Integrating Oracle and Kafka

Robin Moffatt | #ACEsAtHome | @rmoff

$ whoami

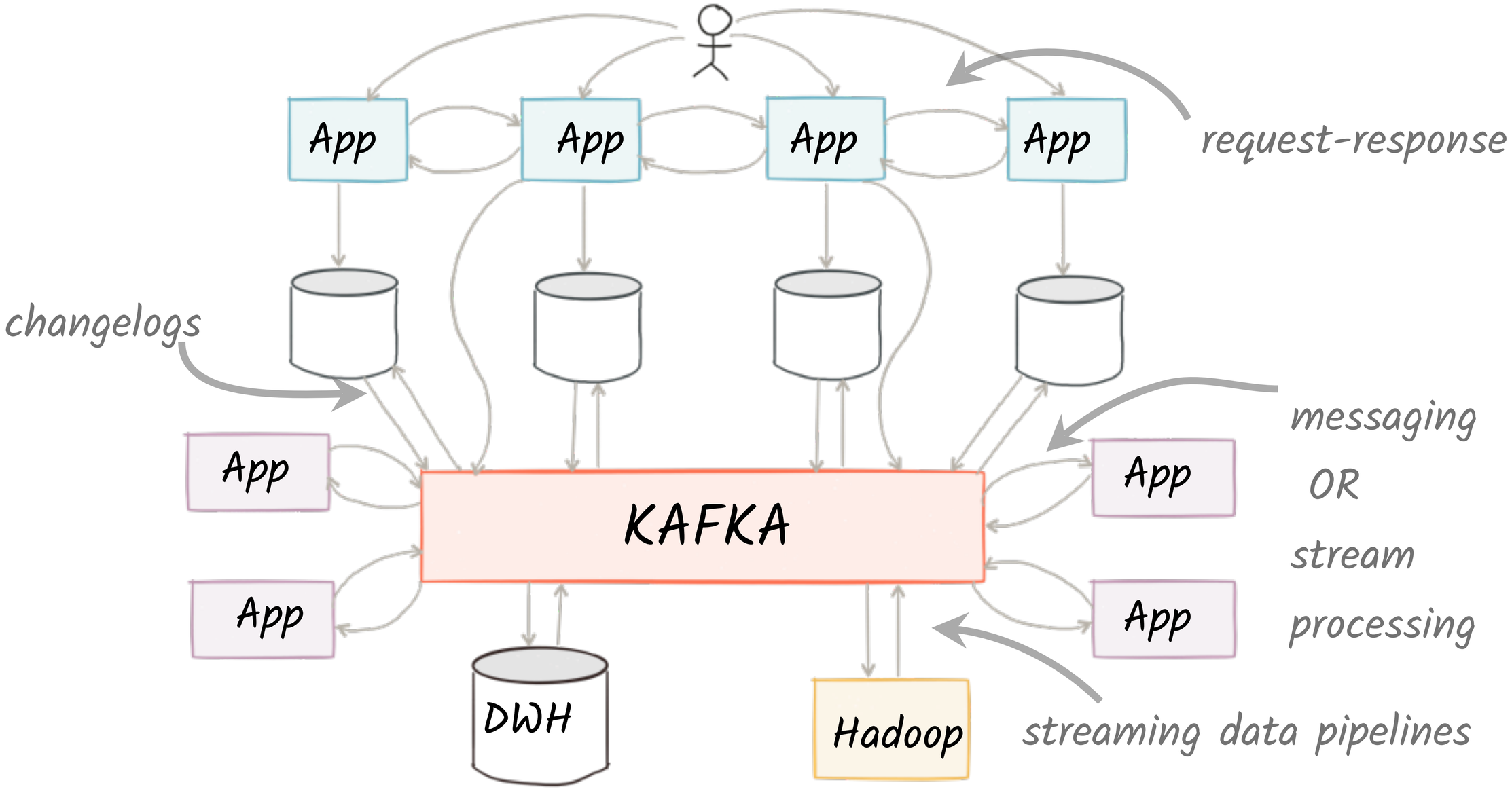
- Robin Moffatt (@rmoff)
- Senior Developer Advocate at Confluent

  (Apache Kafka, not Wikis 😉)

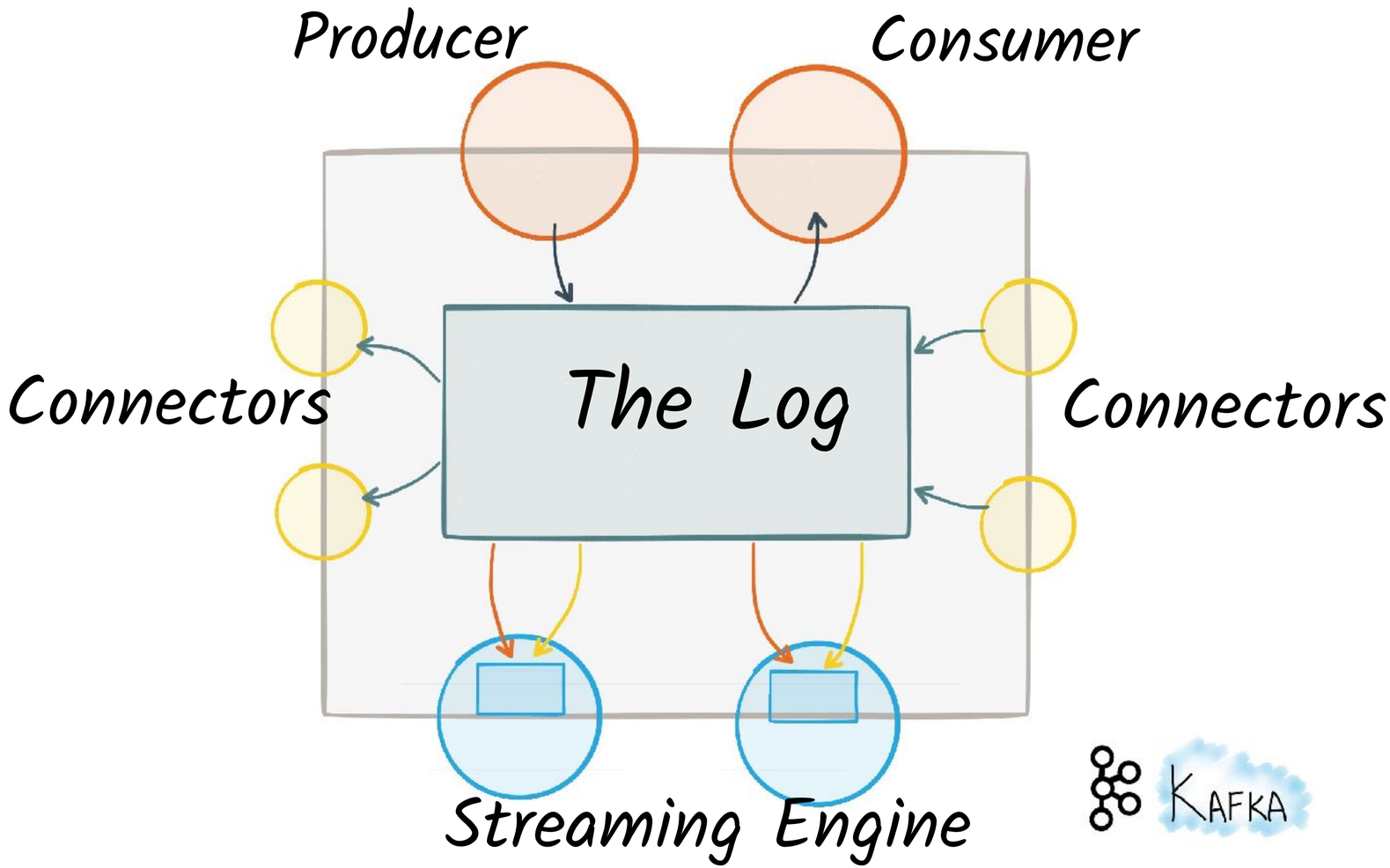
- Working in data & analytics since 2001
- Oracle ACE Director (Alumnus)

http://rmoff.dev/**talks** • http://rmoff.dev/**blog** • http://rmoff.dev/**youtube**

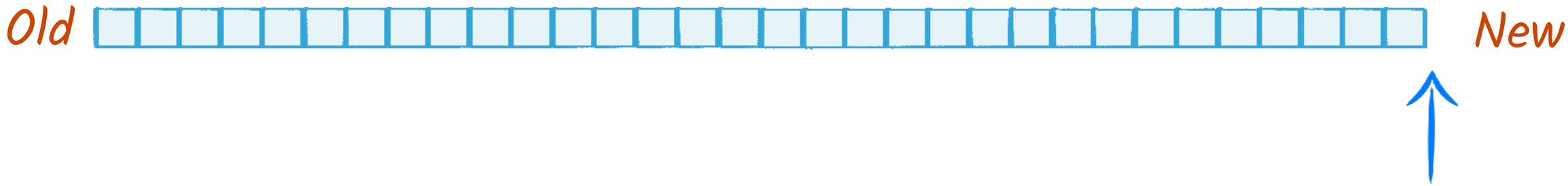
Kafka is an Event Streaming Platform



What is an Event Streaming Platform?



Immutable Event Log



Messages are added at the end of the log

Topics

Clicks

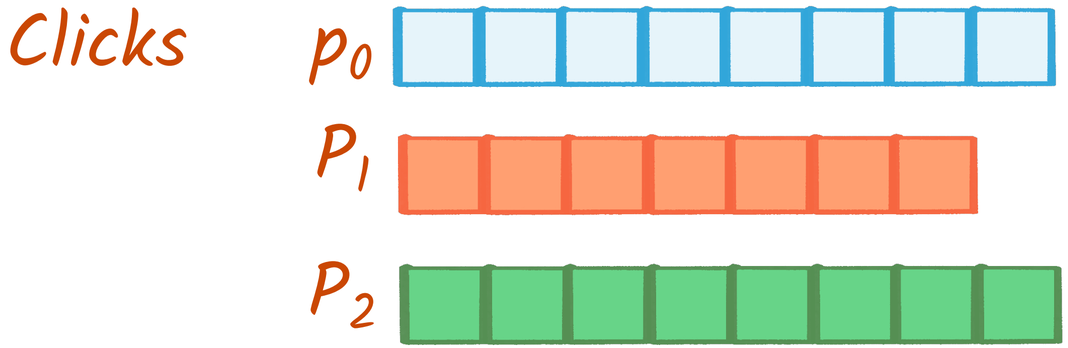
Orders

Customers

Topics are similar in concept

to tables in a database

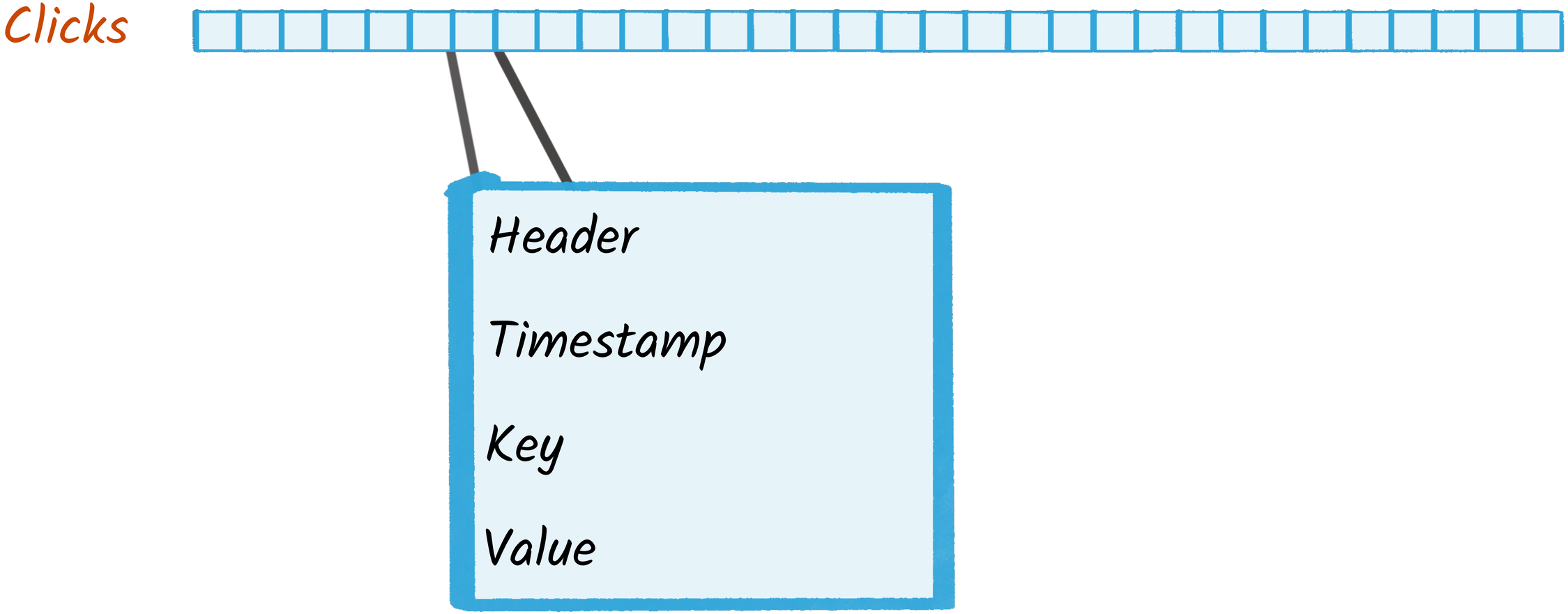
Partitions



Messages are guaranteed to be strictly ordered within a partition

Messages are just KN bytes

plus headers + timestamp



Serialisation & Schemas

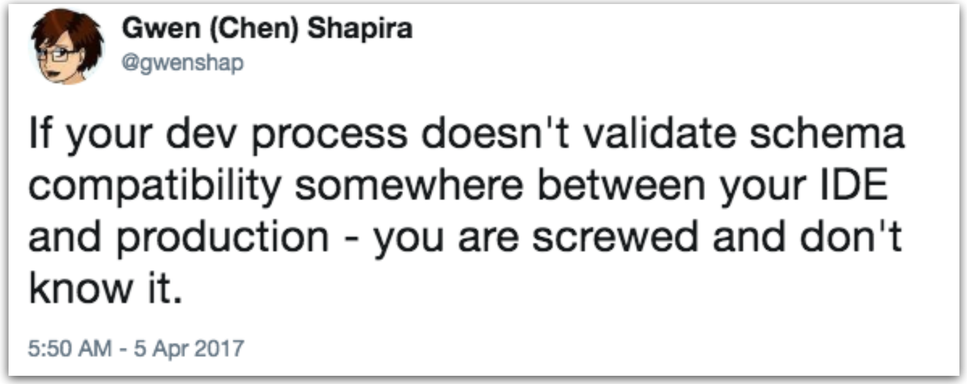
Avro 👍

Protobuf 👍

JSON Schema 👍

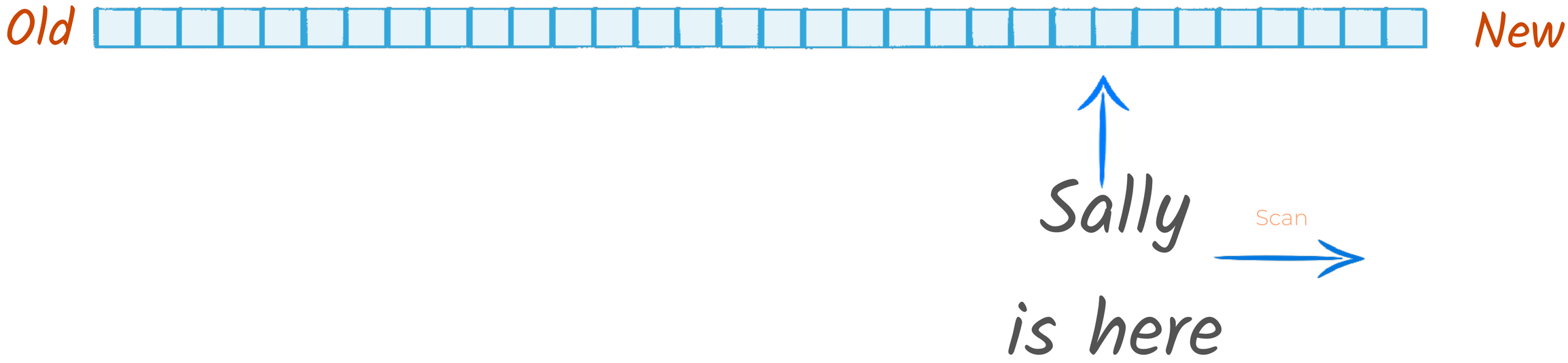
JSON 😬

CSV

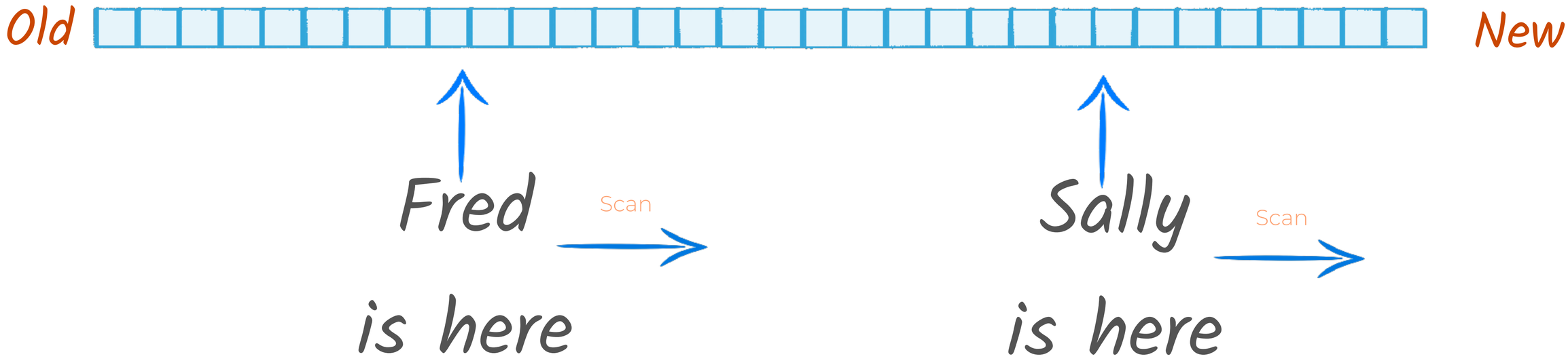


https://qconnewyork.com/system/files/presentation-slides/qcon_17_-_schemas_and_apis.pdf

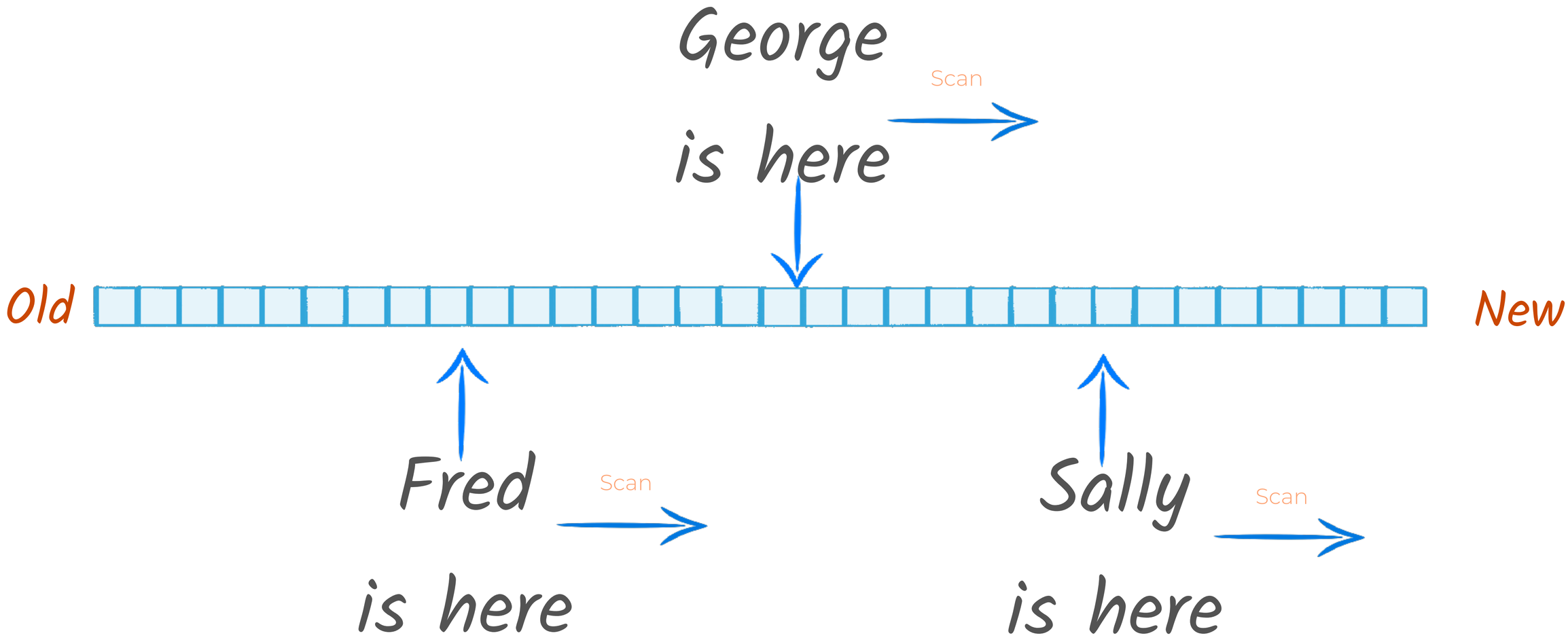
Consumers have a position all of their own



Consumers have a position all of their own

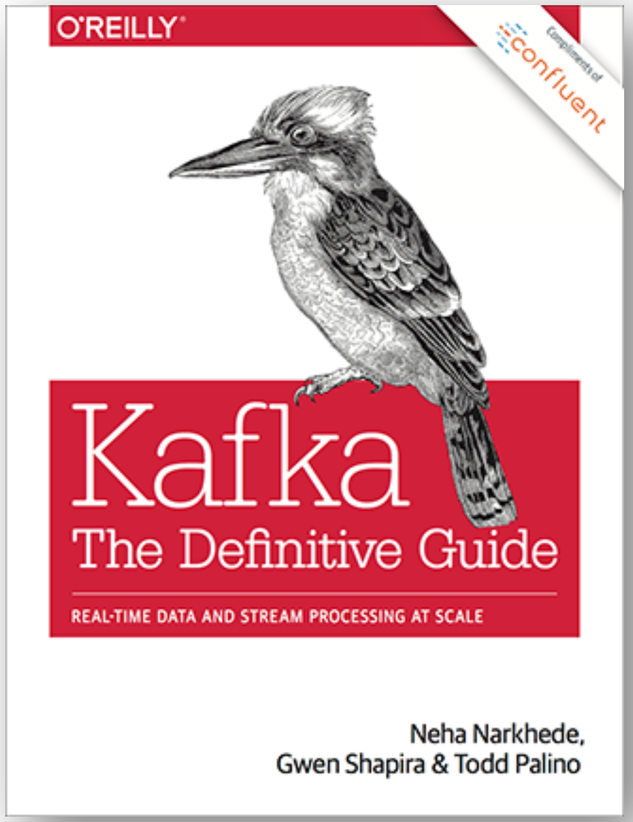


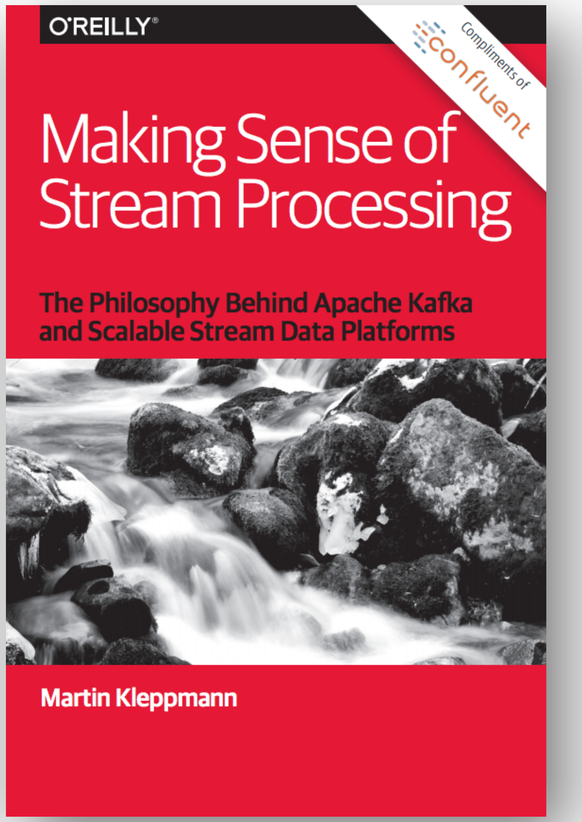
Consumers have a position all of their own

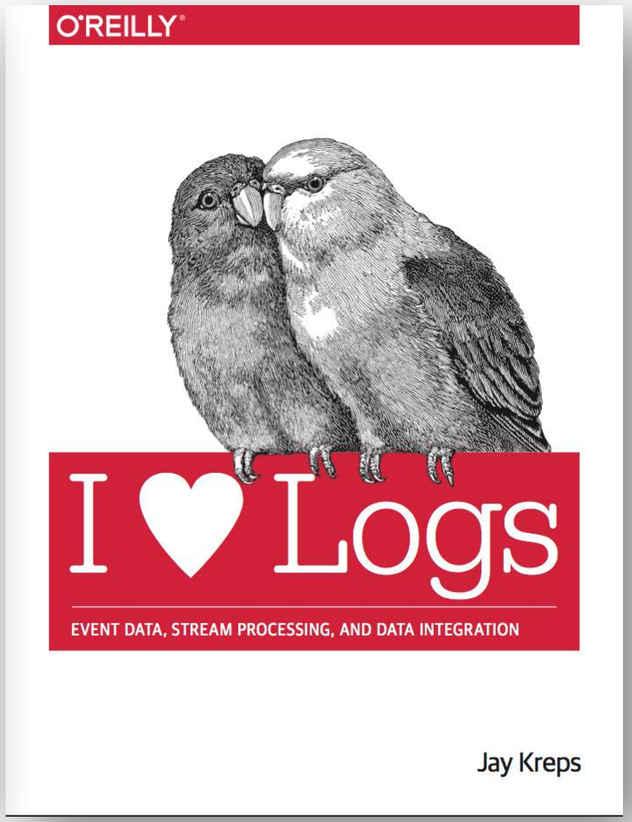


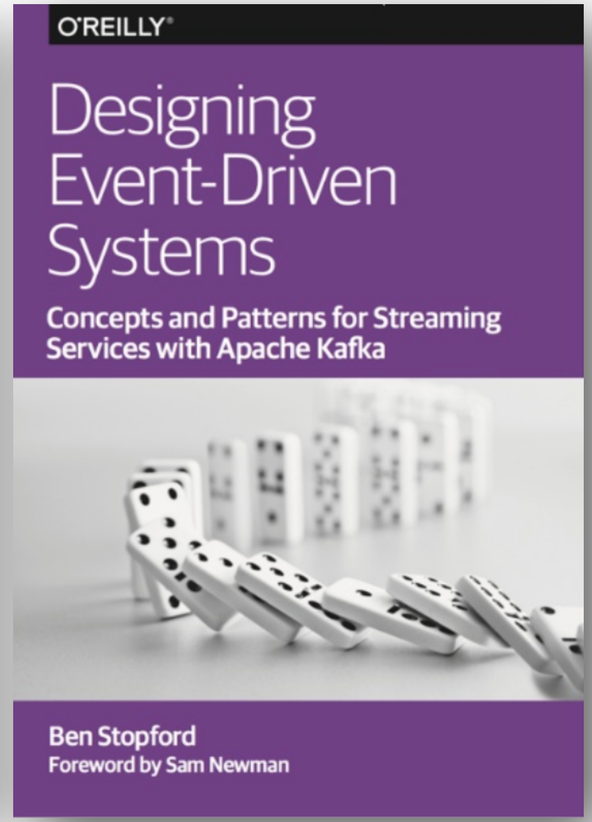
Free Books!

https://rmoff.dev/aces

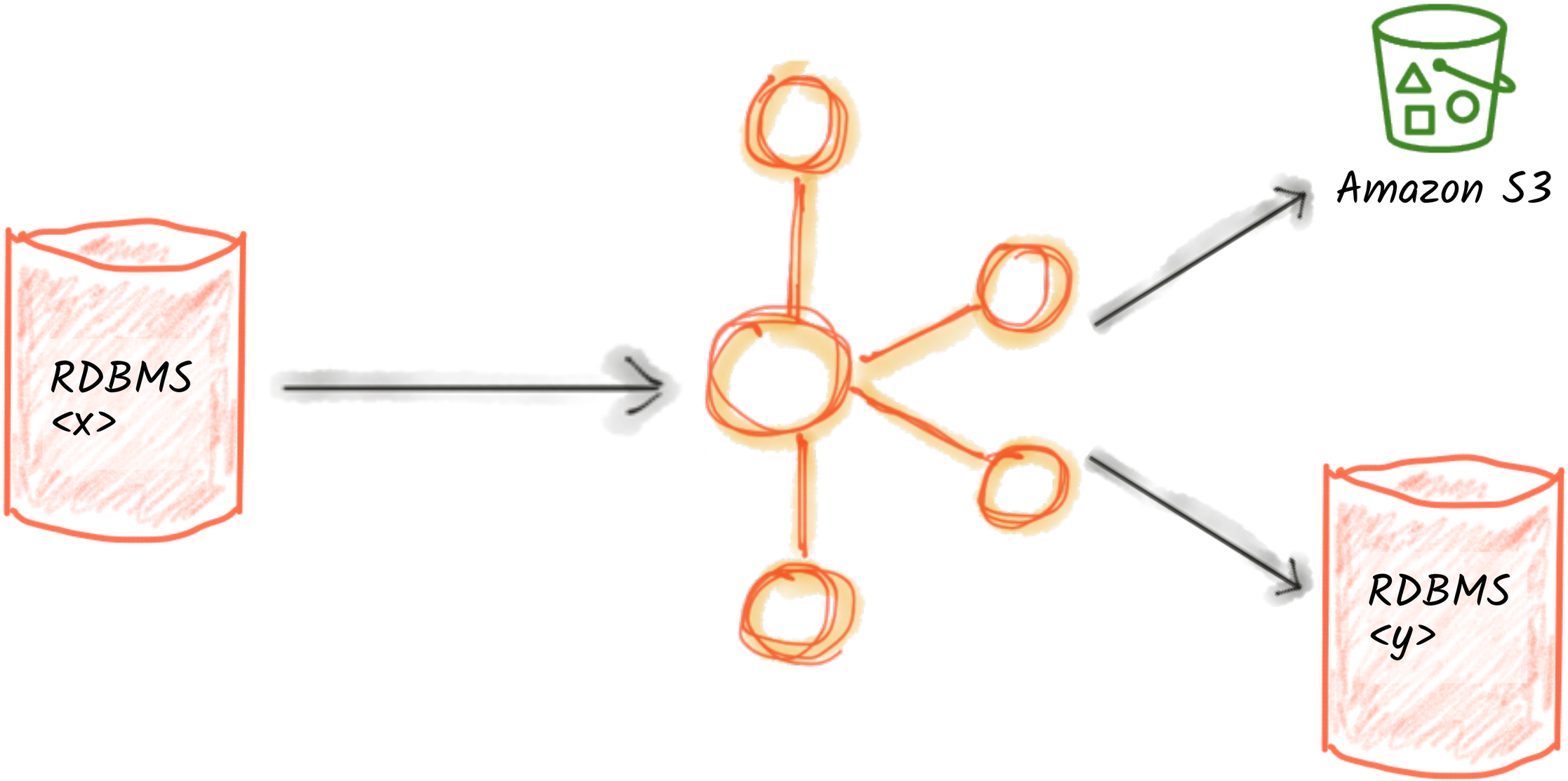




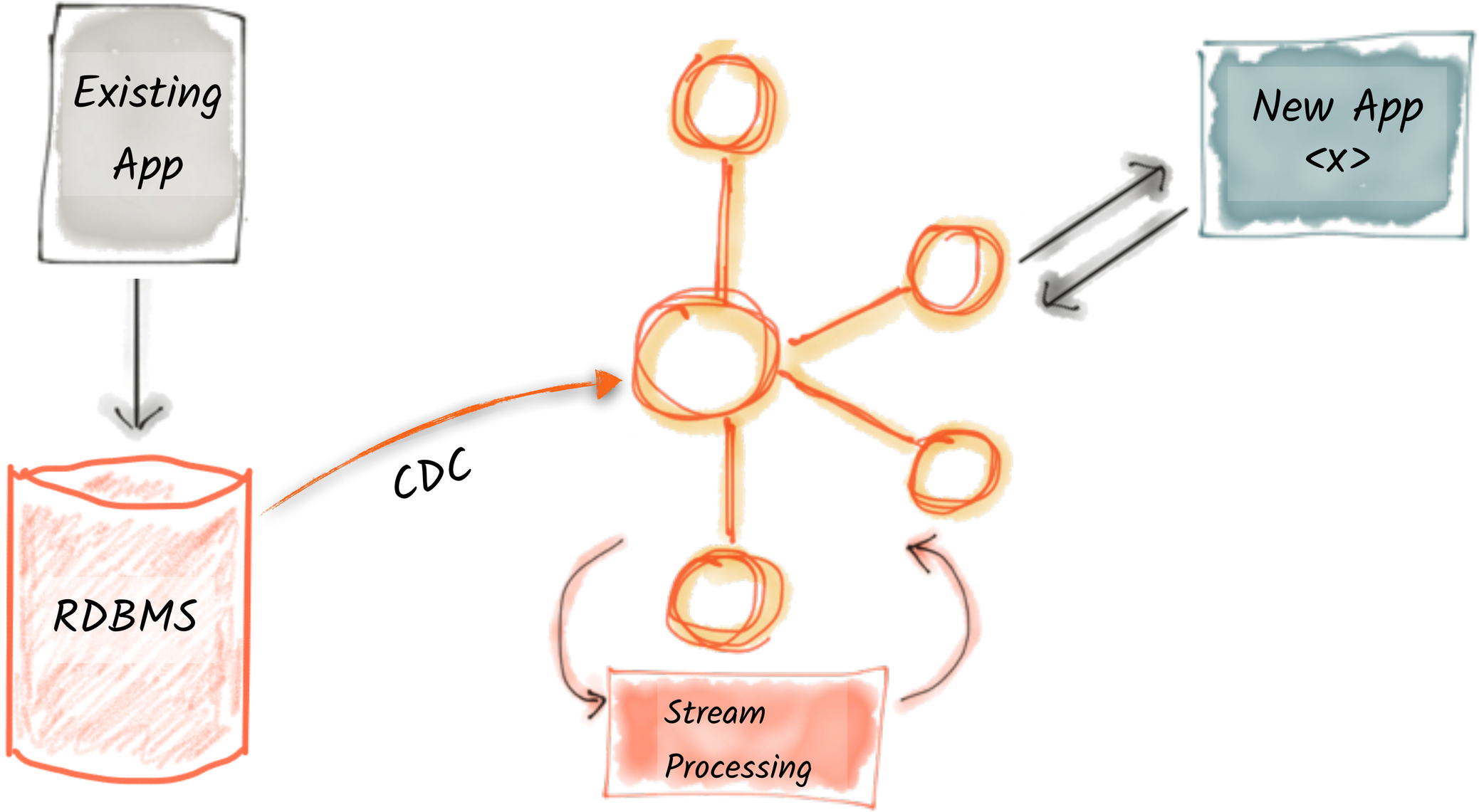




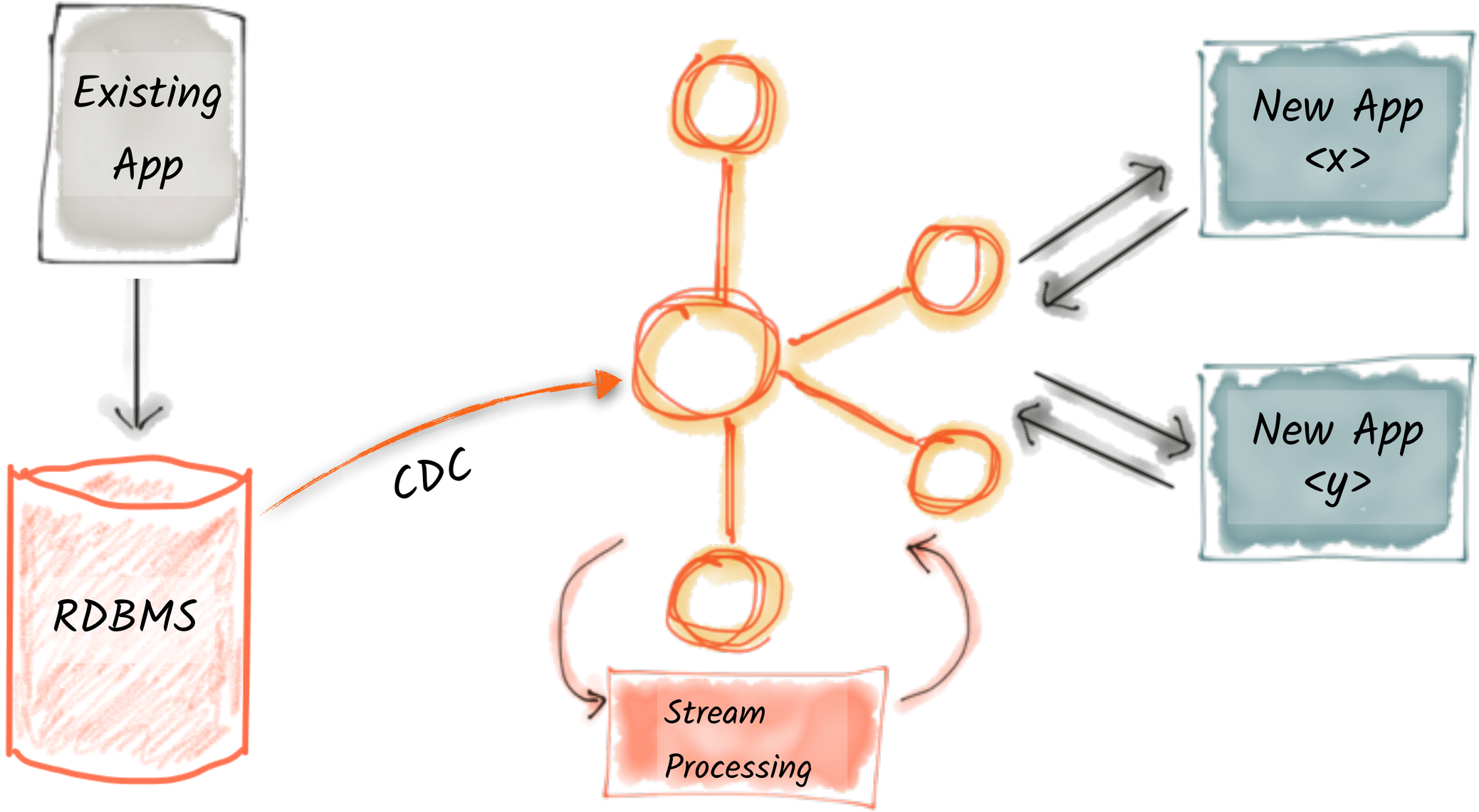
Database Offload



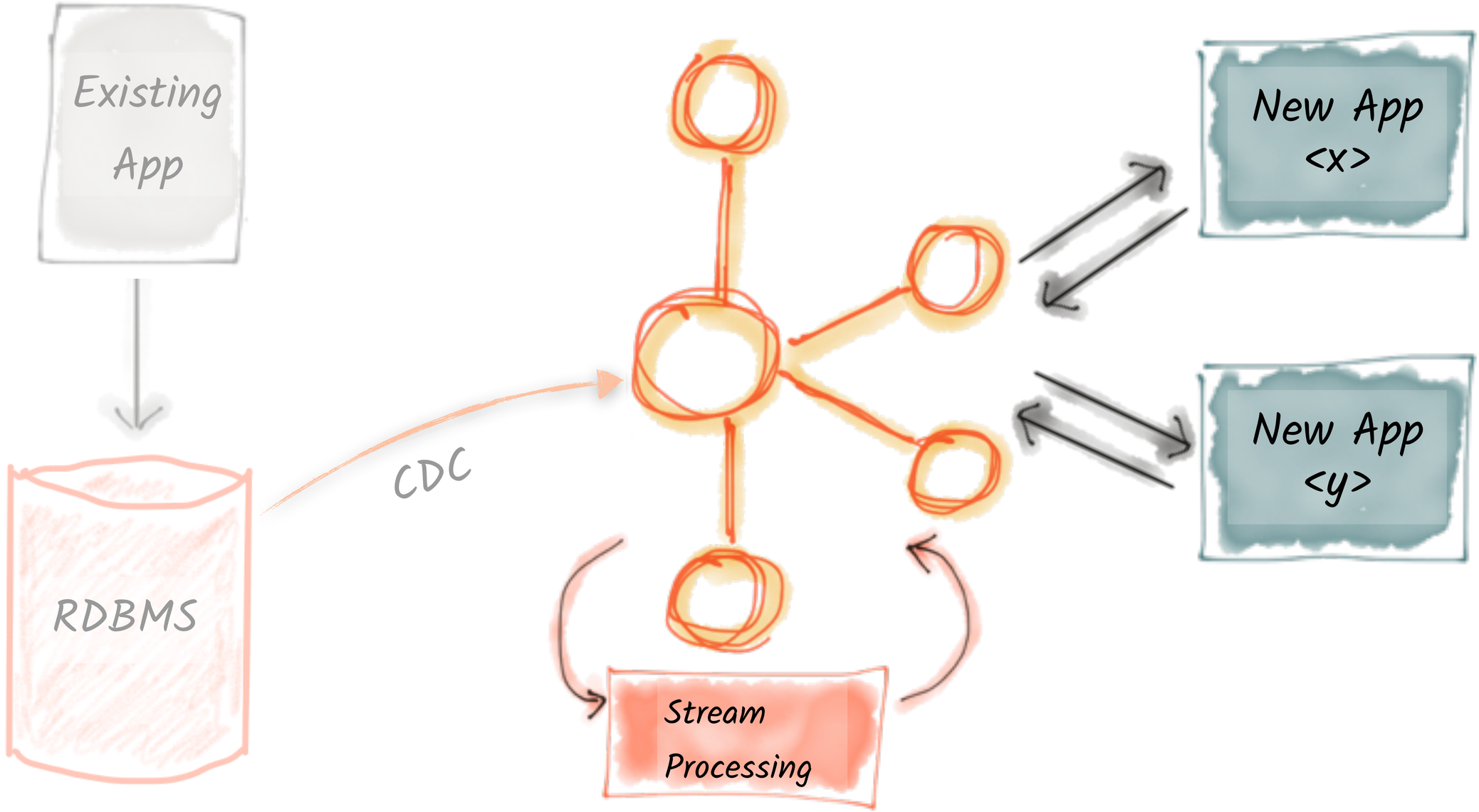
Evolve processing from old systems to new



Evolve processing from old systems to new

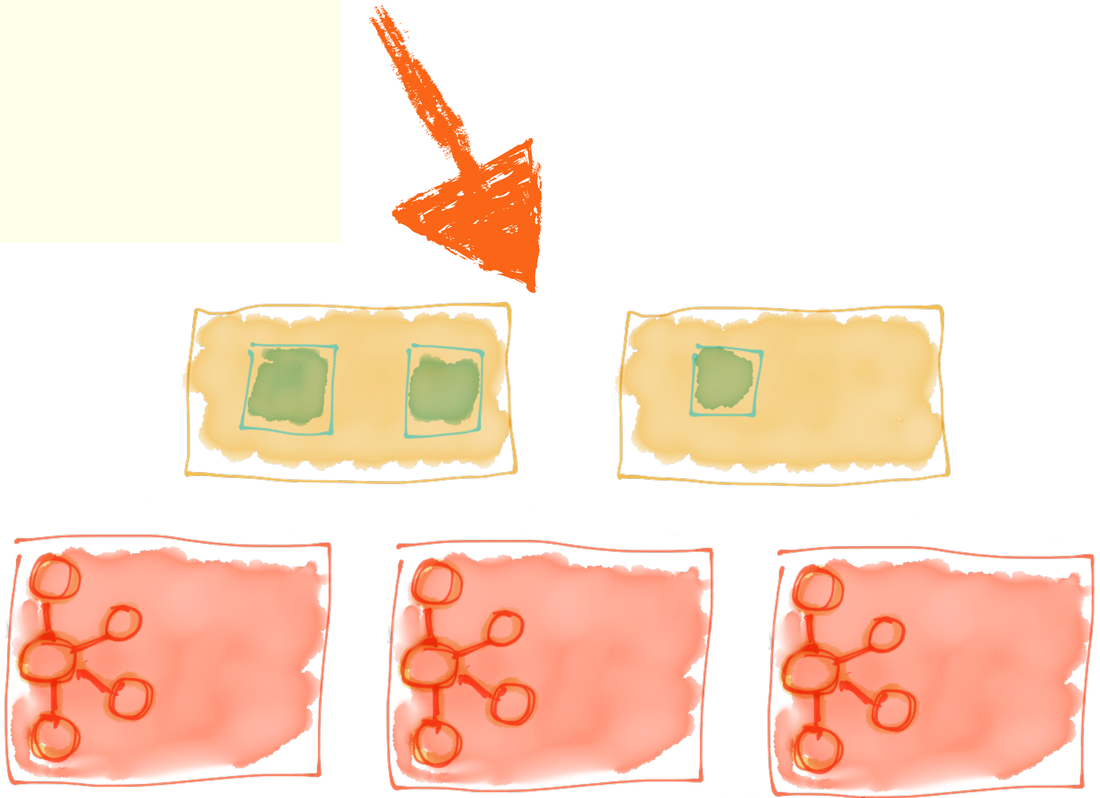


Evolve processing from old systems to new



Streaming Integration with Kafka Connect

Sources



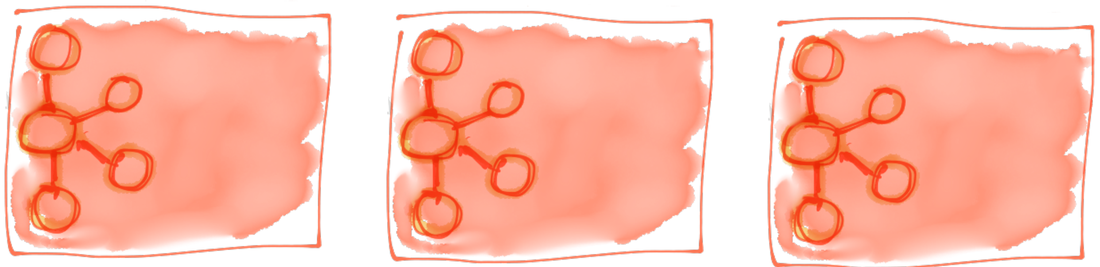
Kafka Connect

Kafka Brokers

Streaming Integration with Kafka Connect

Sinks

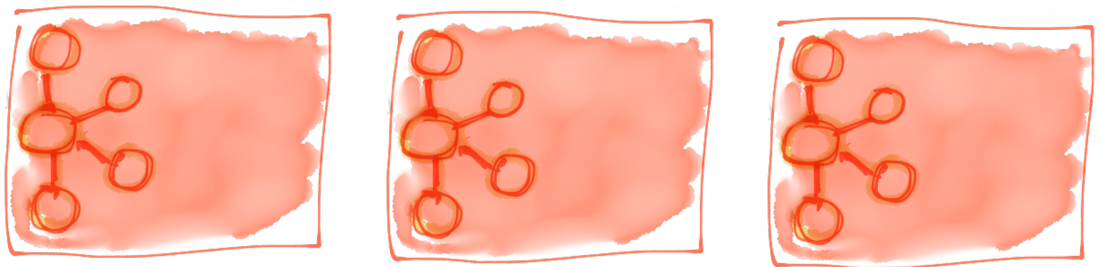
Kafka Connect



Kafka Brokers

Streaming Integration with Kafka Connect

Kafka Connect



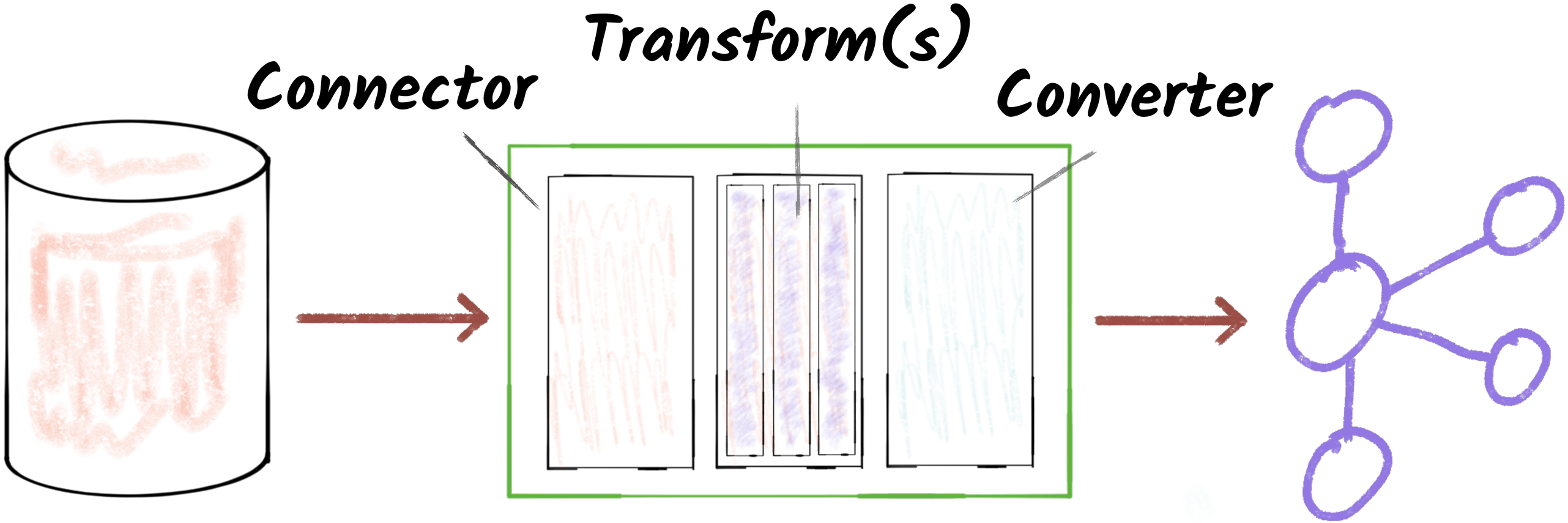
Kafka Brokers

Look Ma, No Code!

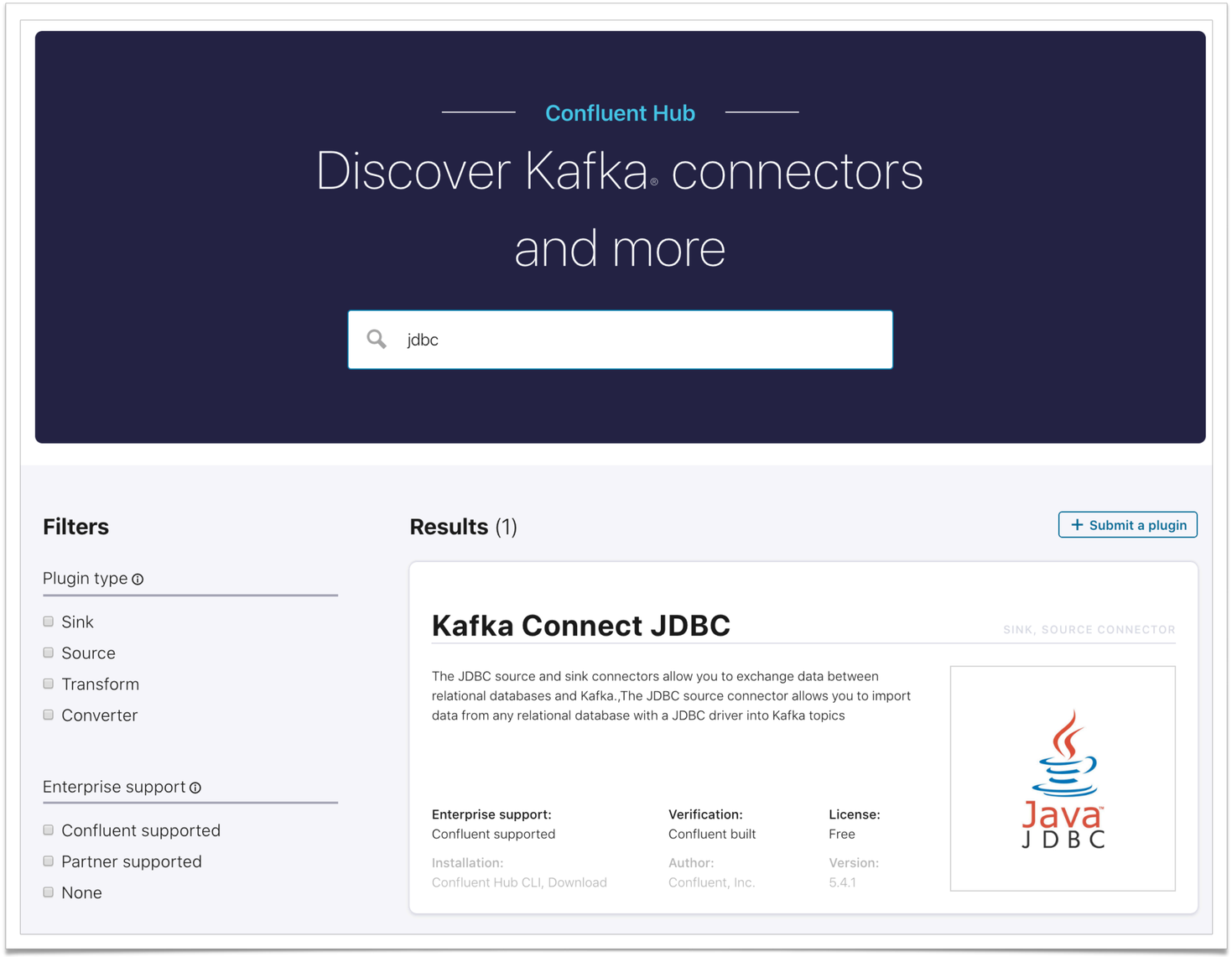
```
{
  "connector.class":
    "io.confluent.connect.jdbc.JdbcSourceConnector",
  "connection.url":
    "jdbc:mysql://asgard:3306/demo",
  "table.whitelist":
    "sales,orders,customers"
}
```

https://docs.confluent.io/current/connect/

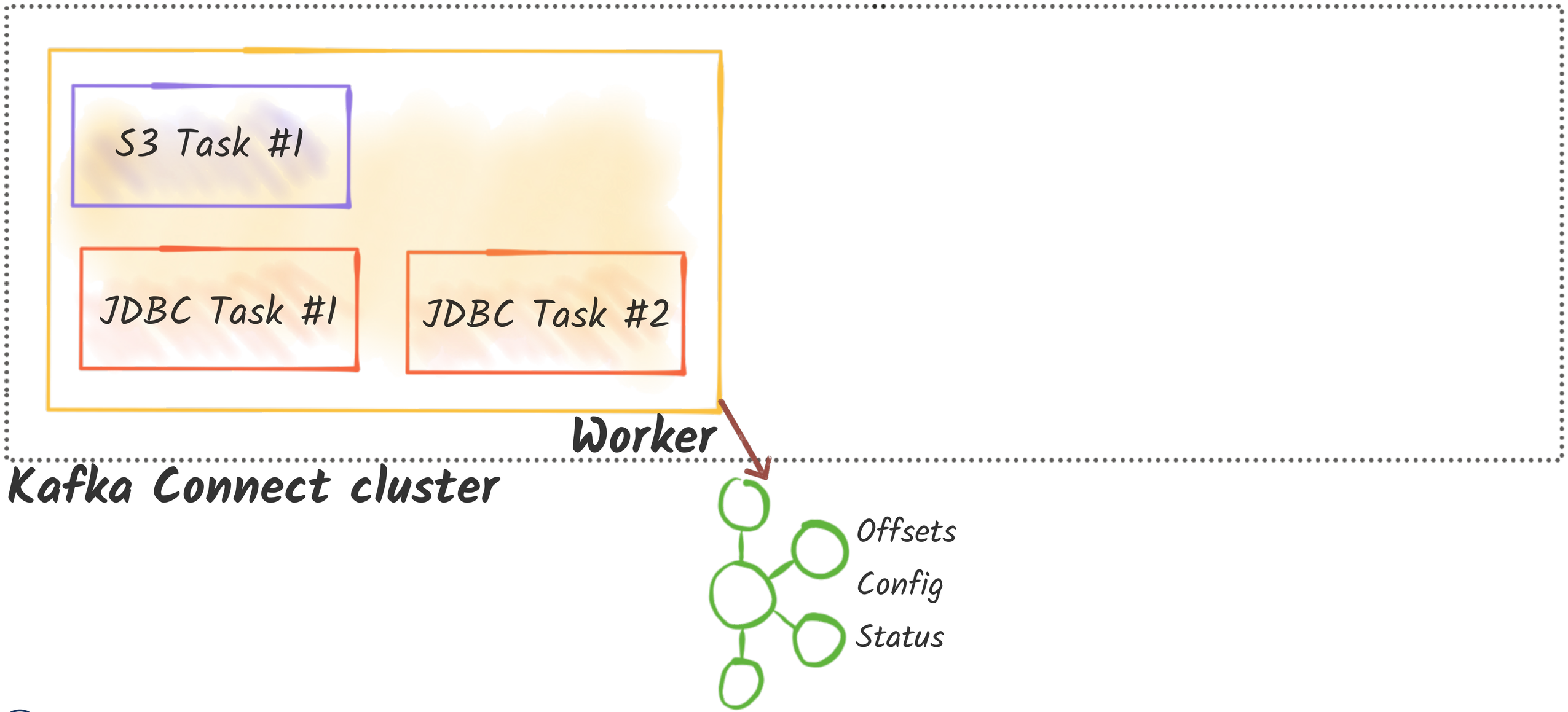
Extensible



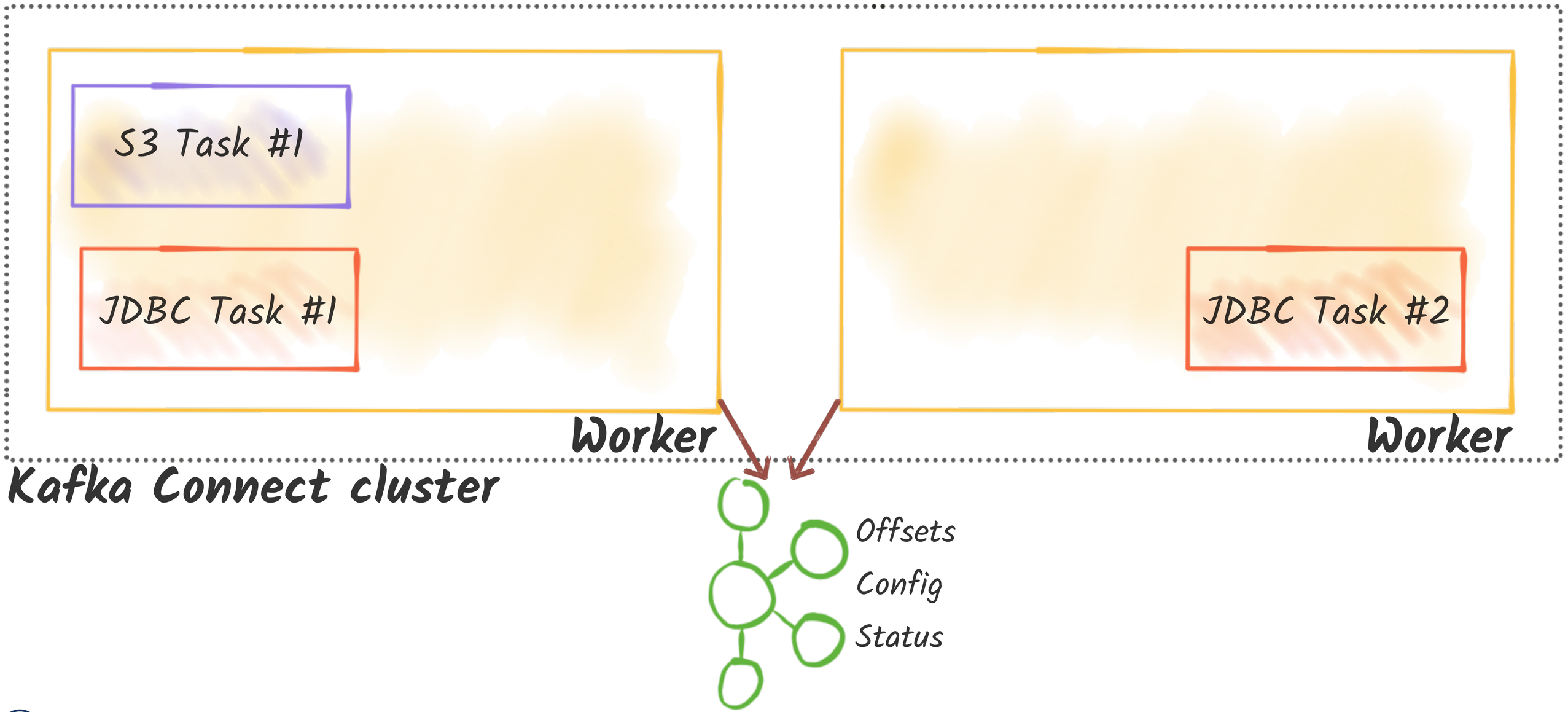
hub.confluent.io



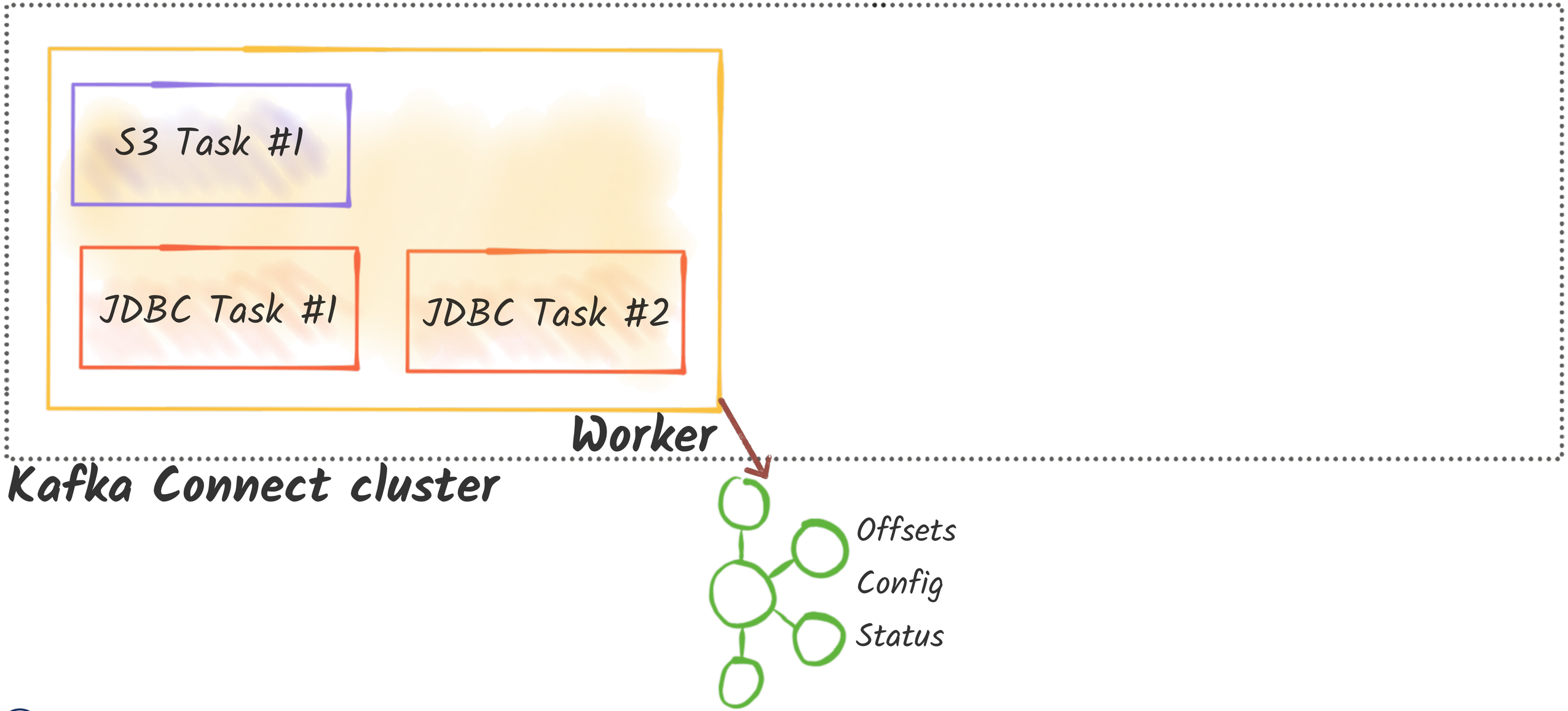
Single Kafka Connect node



Kafka Connect - scalable and fault-tolerant



Automatic fault tolerance



Change-Data-Capture (CDC)

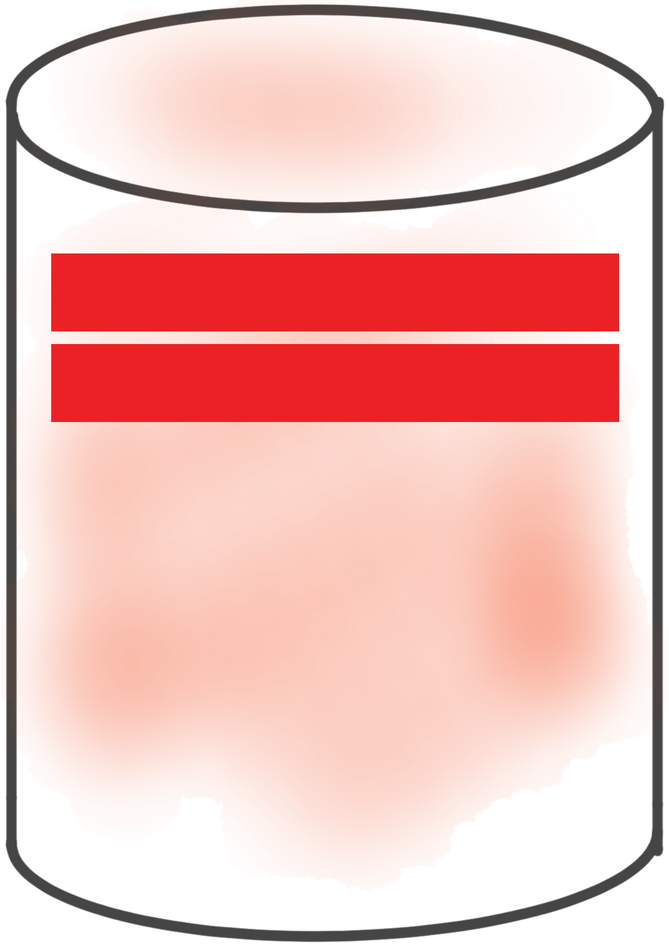
Query-based

Log-based

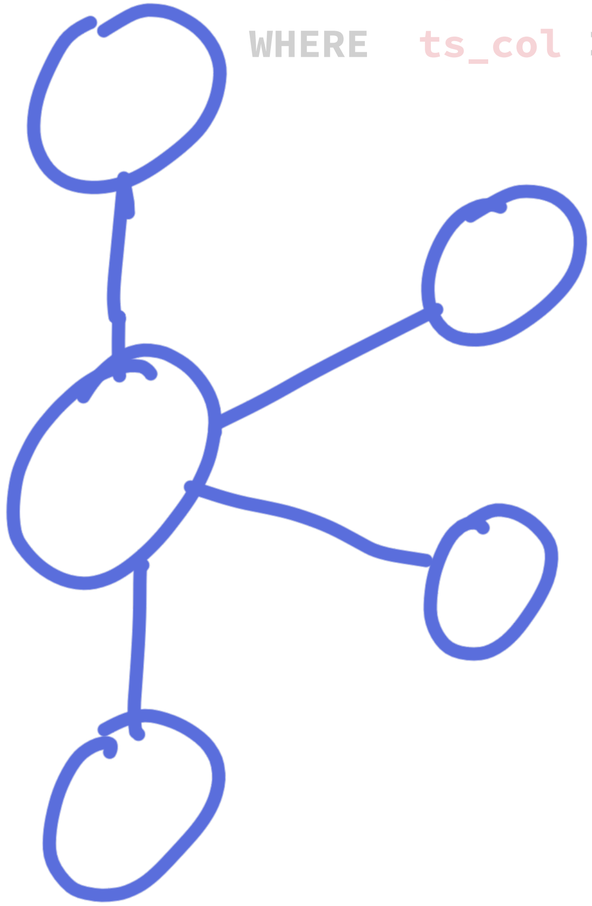
Query-based CDC

```
SELECT *
FROM   my_table
WHERE  ts_col > previous ts
```

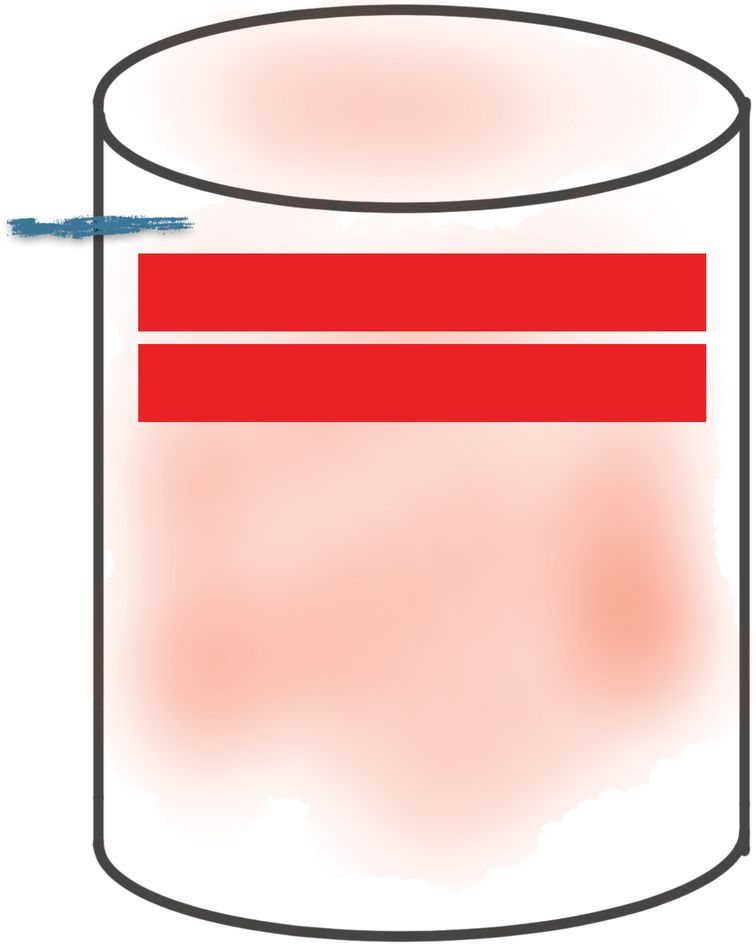
Query-based CDC



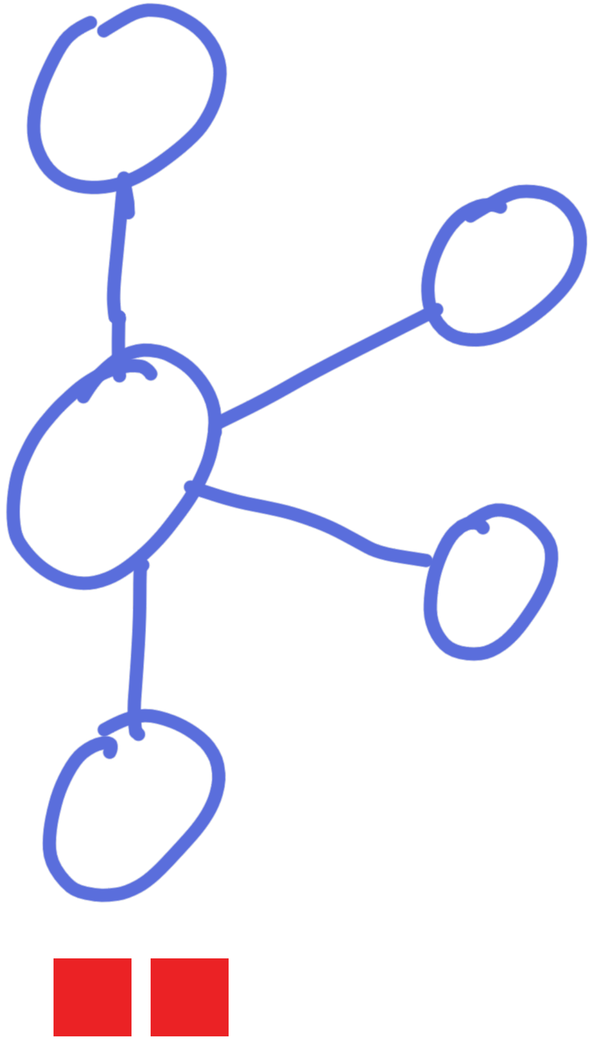
```
SELECT *
FROM   my_table
WHERE  ts_col > previous ts
```



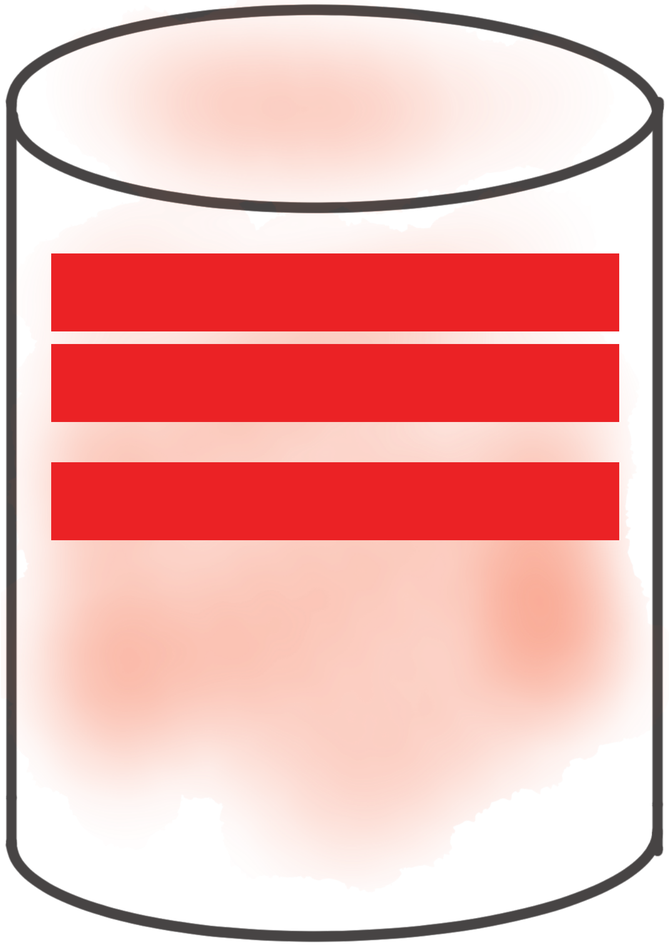
Query-based CDC



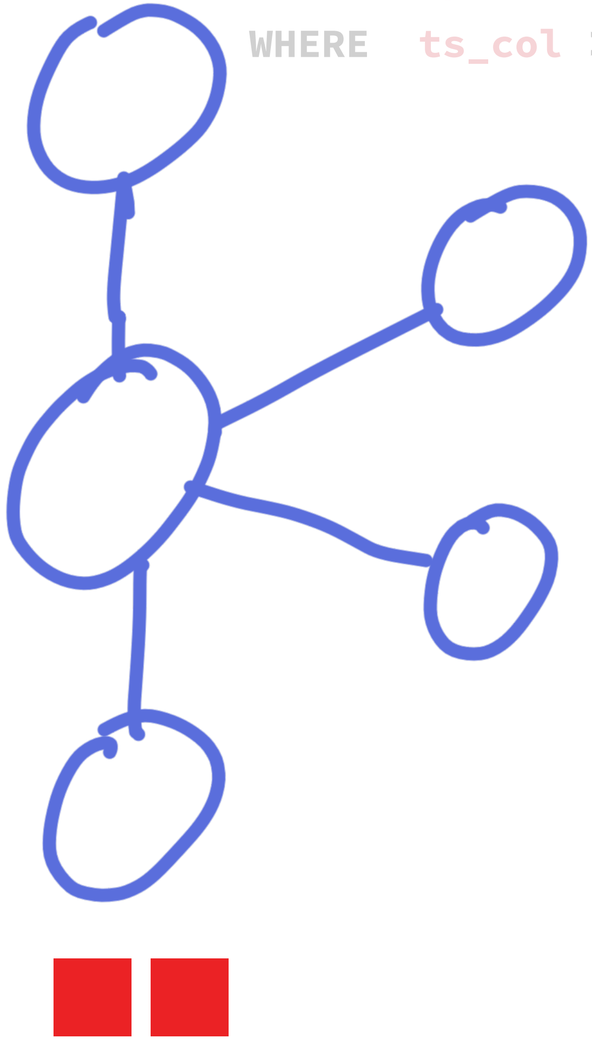
```
SELECT *
FROM   my_table
WHERE  ts_col >
         previous ts
```



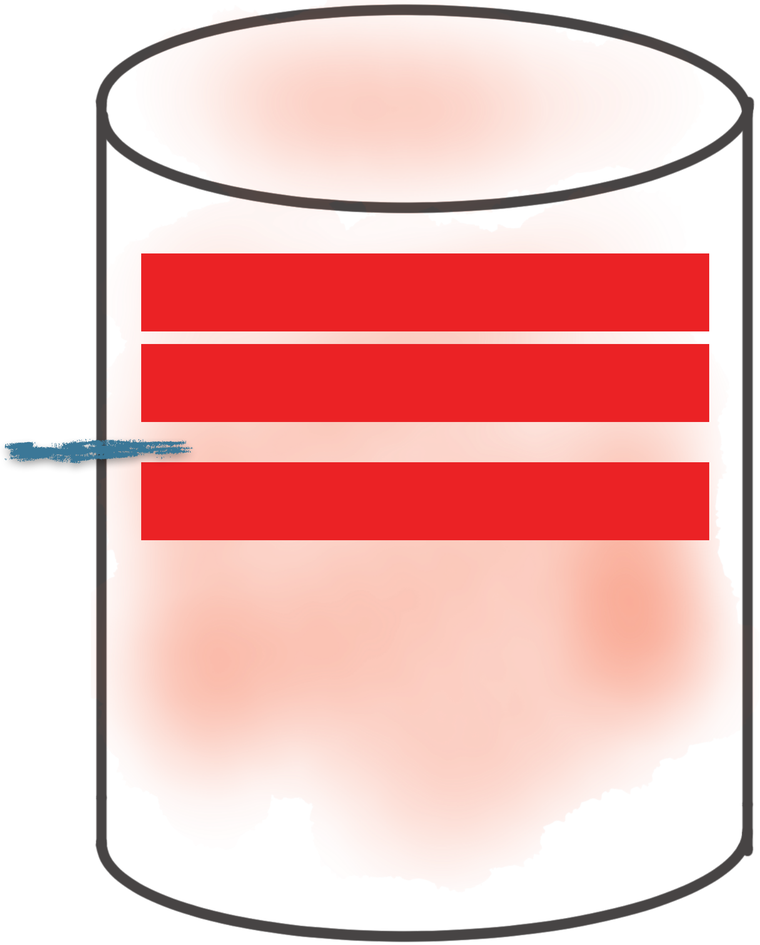
Query-based CDC



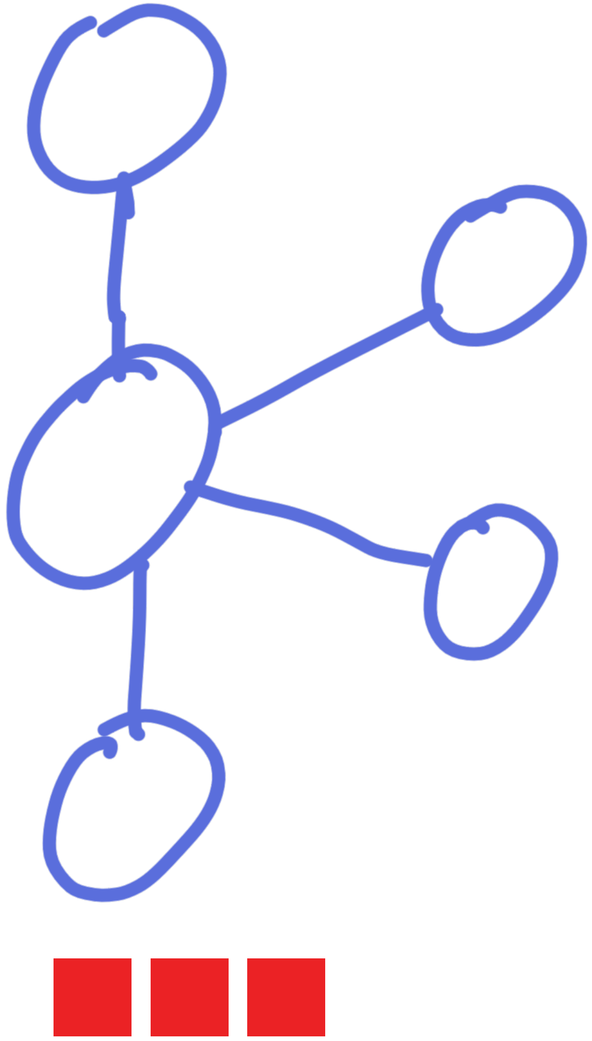
```
SELECT *
FROM   my_table
WHERE  ts_col > previous ts
```



Query-based CDC



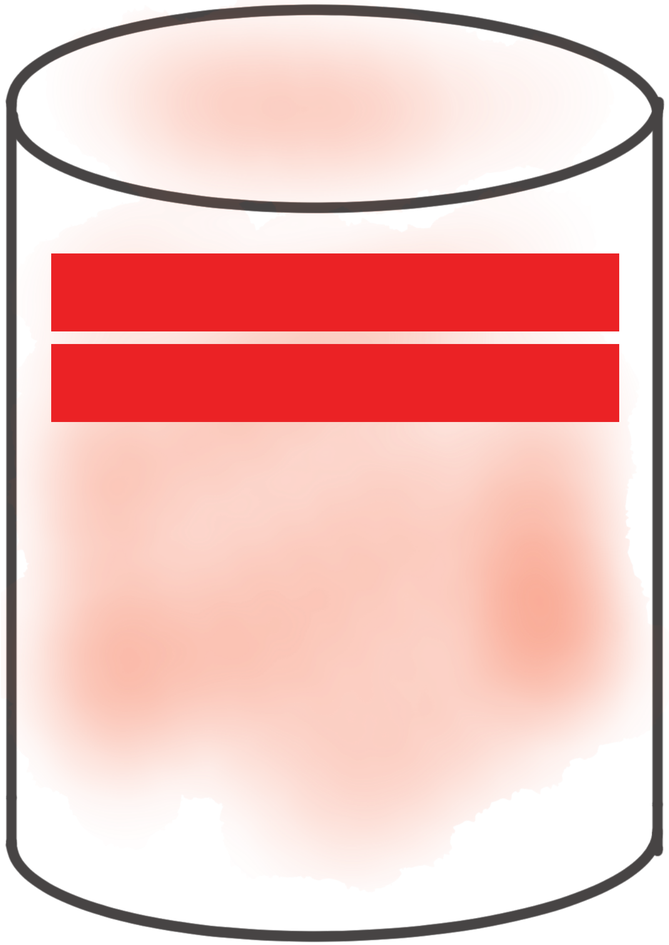
```
SELECT *
FROM   my_table
WHERE  ts_col >
          previous ts
```

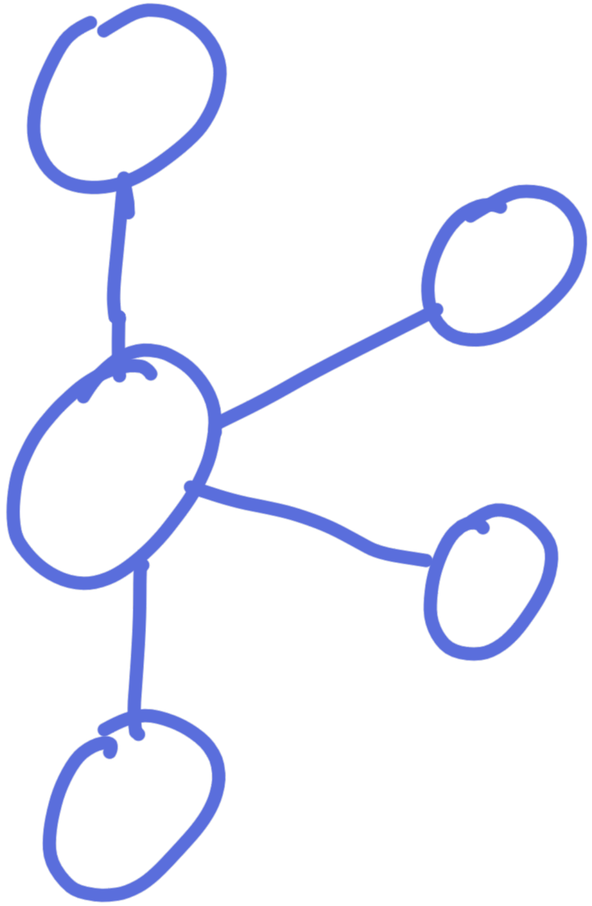


Log-based CDC

```
#051024 17:24:13 server id 1  end_log_pos 98
# Position  Timestamp   Type   Master ID        Size      Master Pos    Flags
# 00000004 9d fc 5c 43   0f   01 00 00 00   5e 00 00 00   62 00 00 00   00 00
# 00000017 04 00 35 2e 30 2e 31 35  2d 64 65 62 75 67 2d 6c |..5.0.15.debug.l|
# 00000027 6f 67 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |og..............|
# 00000037 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
# 00000047 00 00 00 00 9d fc 5c 43  13 38 0d 00 08 00 12 00 |.......C.8......|
# 00000057 04 04 04 04 12 00 00 4b  00 04 1a               |.......K...|
```

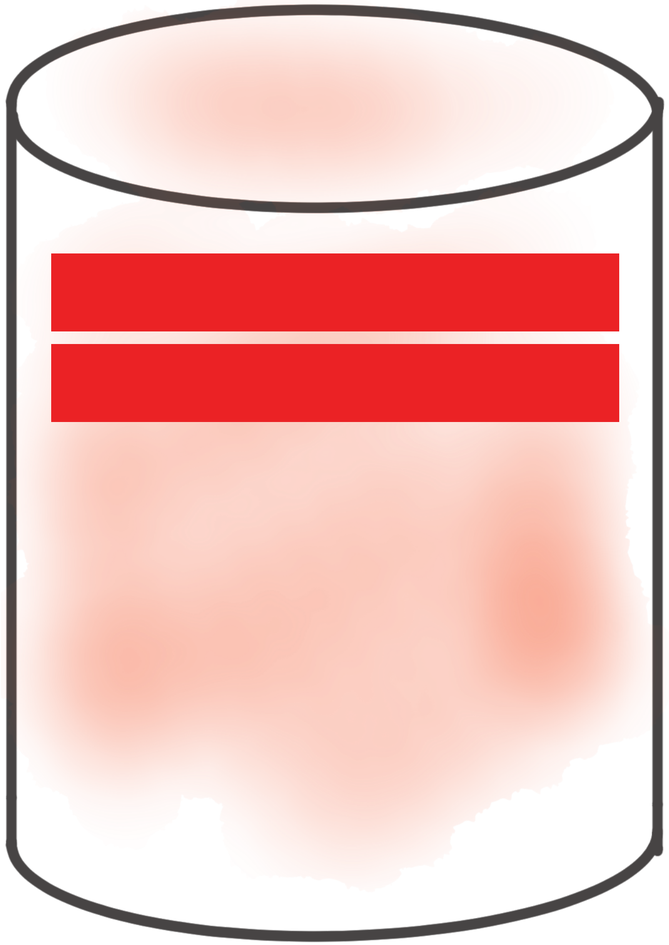
Log-based CDC

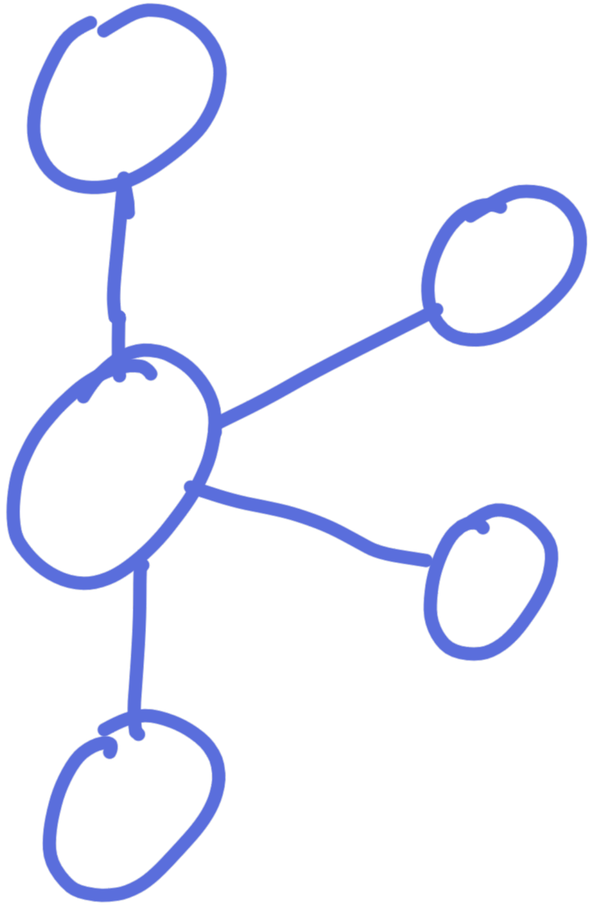




Transaction log

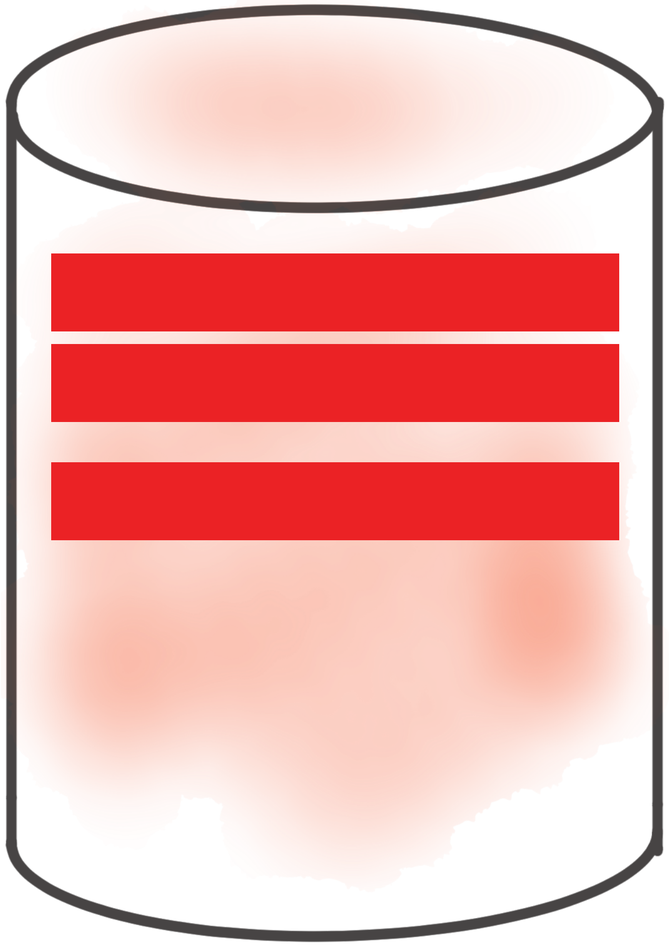
Log-based CDC

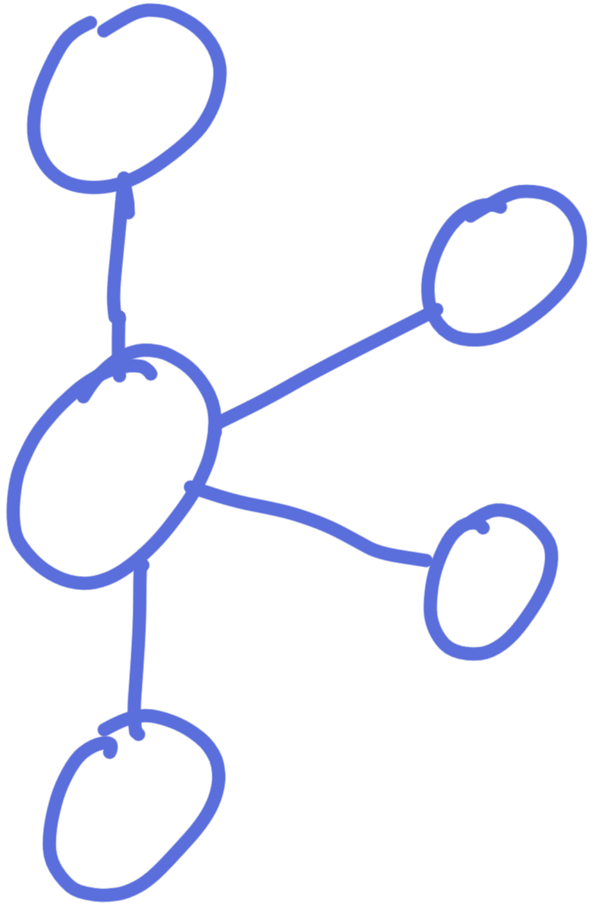




Transaction log

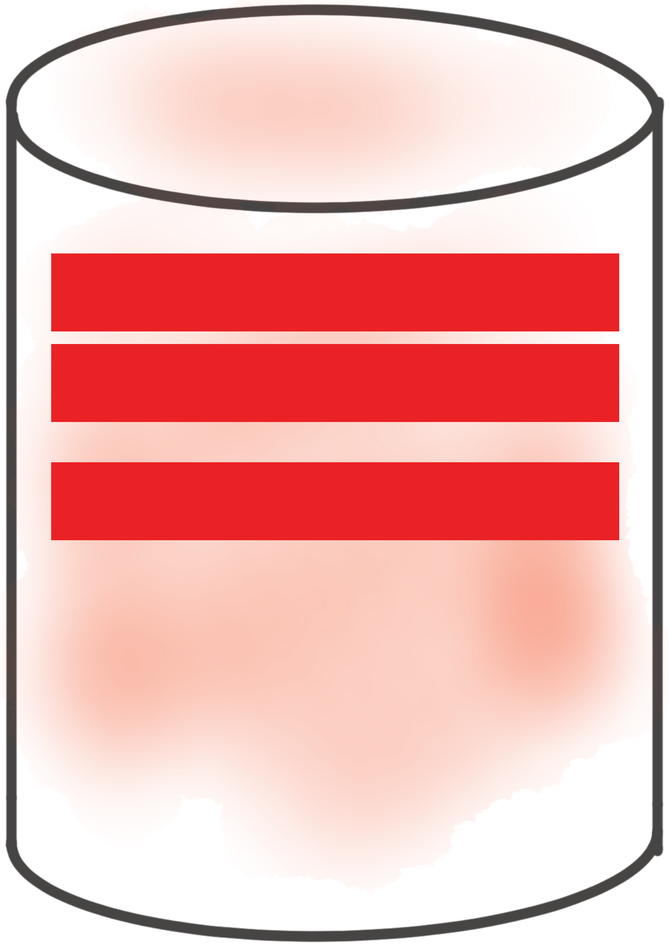
Log-based CDC

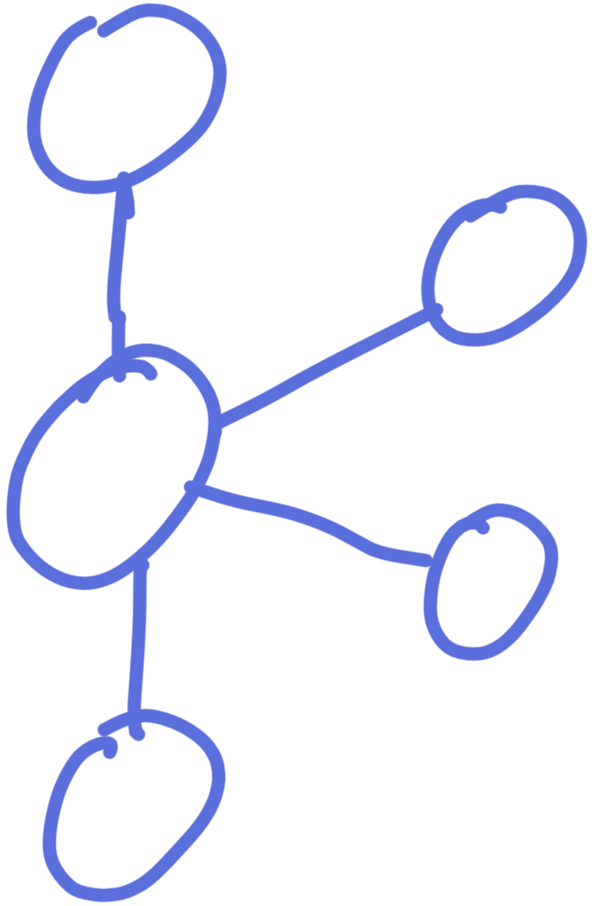




Transaction log

Log-based CDC





Transaction log

Photo by Tom Moffatt

DEMO

@rmoff | #ACEsAtHome | @confluentinc

Query-based CDC
Log-based CDC
JAKE-CLARK.TUMBLR
imgflip.com

Query-based CDC

- ✅ Usually easier to setup
- ✅ Requires fewer permissions
- 🛑 Needs specific columns in source schema to track changes
- 🛑 Impact of polling the DB (or higher latencies tradeoff)
- 🛑 Can't track deletes
- 🛑 Can't track multiple events between polling interval

Read more: http://cnfl.io/kafka-c

Photo by Matese Fields on Unsplash

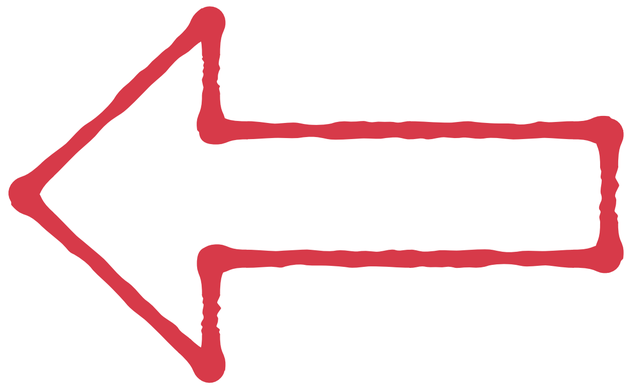
Query-based CDC

```
SELECT *
FROM   my_table
WHERE  ts_col > previous ts
```

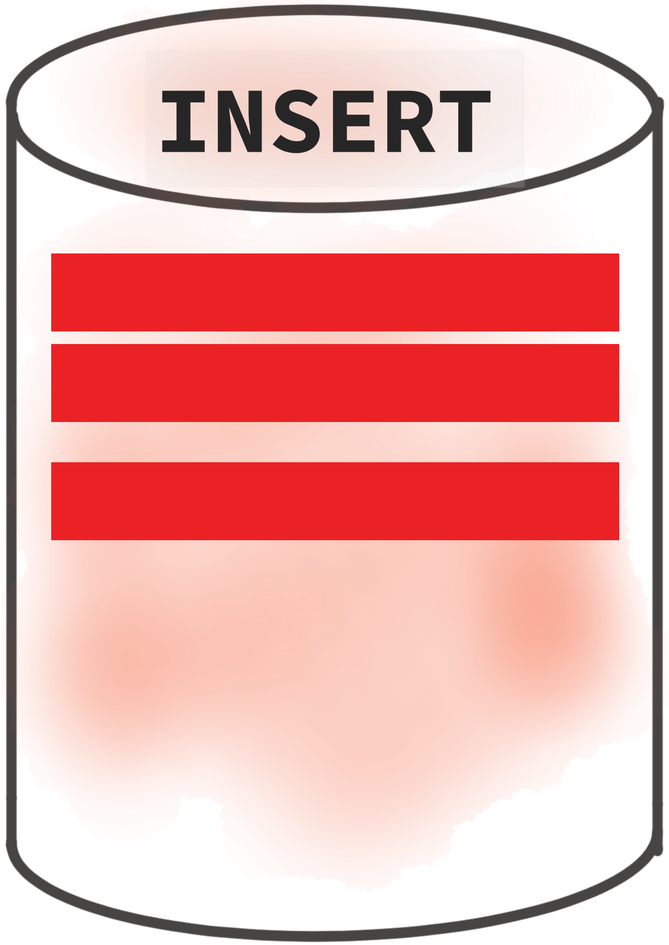
Query-based CDC

```
CREATE TABLE my_table (
  ID INT,
  FOO VARCHAR,
  BAR VARCHAR,
  WIBBLE VARCHAR,
  TS_COL TIMESTAMP
    DEFAULT CURRENT_TIMESTAMP
    ON UPDATE CURRENT_TIMESTAMP
```

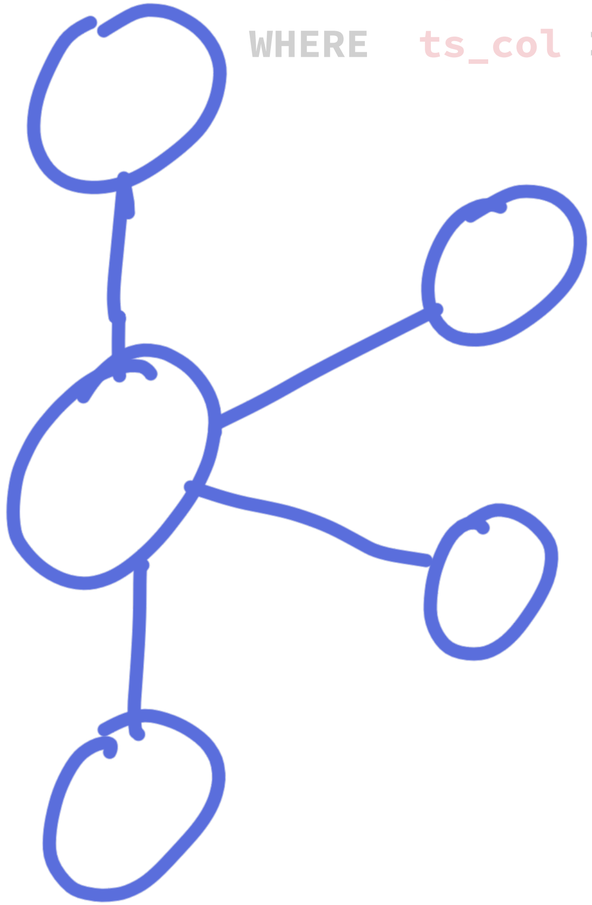
```
SELECT *
FROM   my_table
WHERE  ts_col > previous ts
```



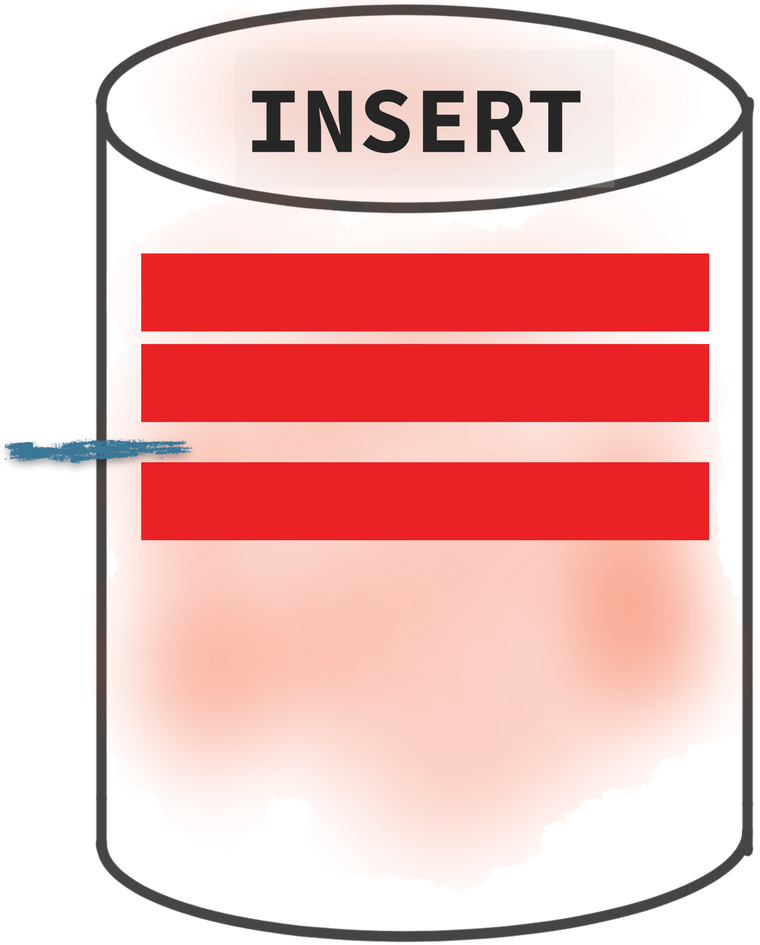
Query-based CDC



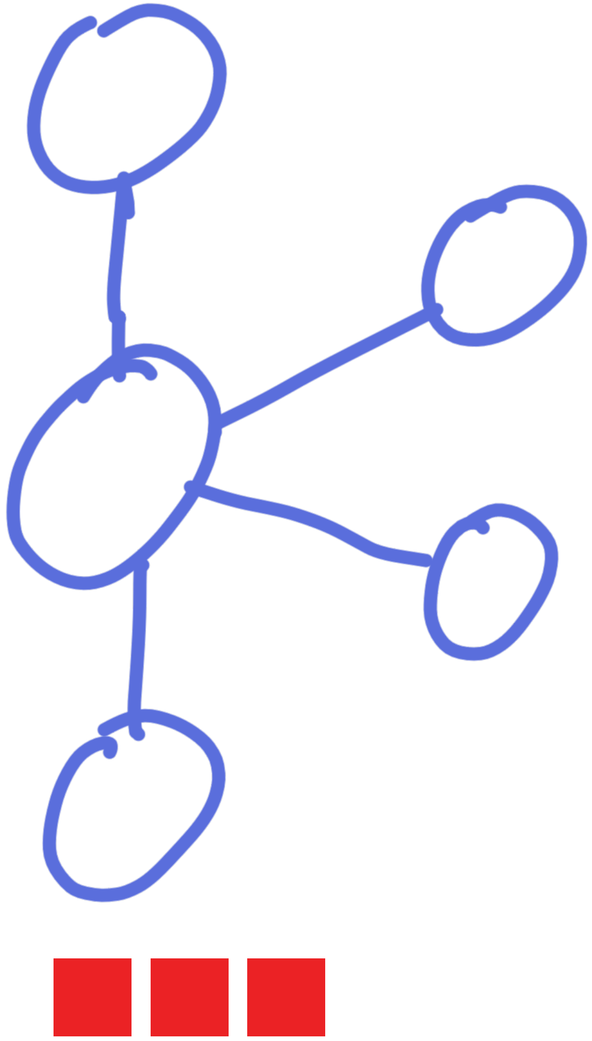

```
SELECT *
FROM   my_table
WHERE  ts_col > previous ts
```



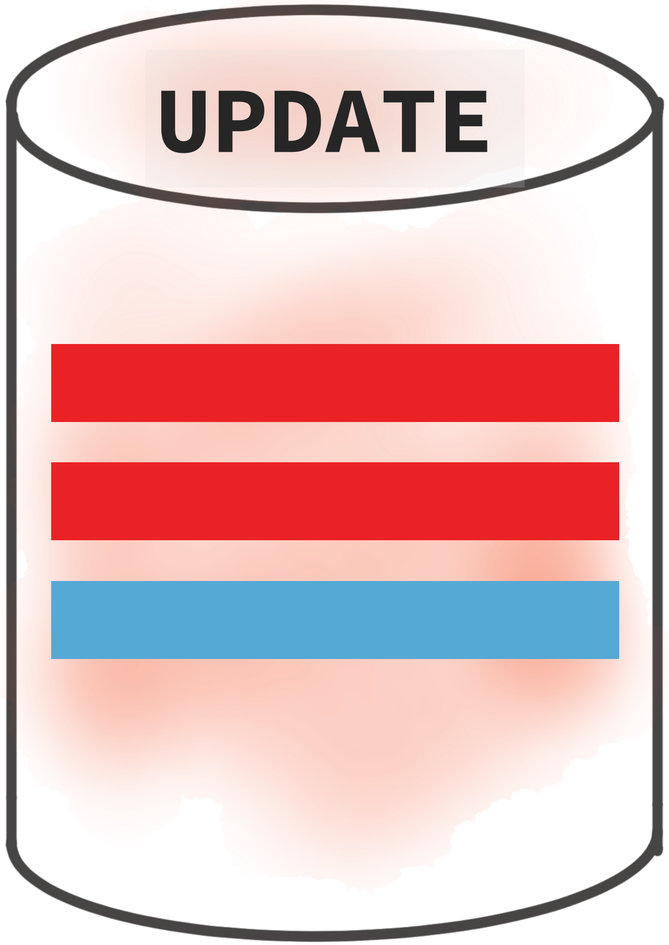
Query-based CDC

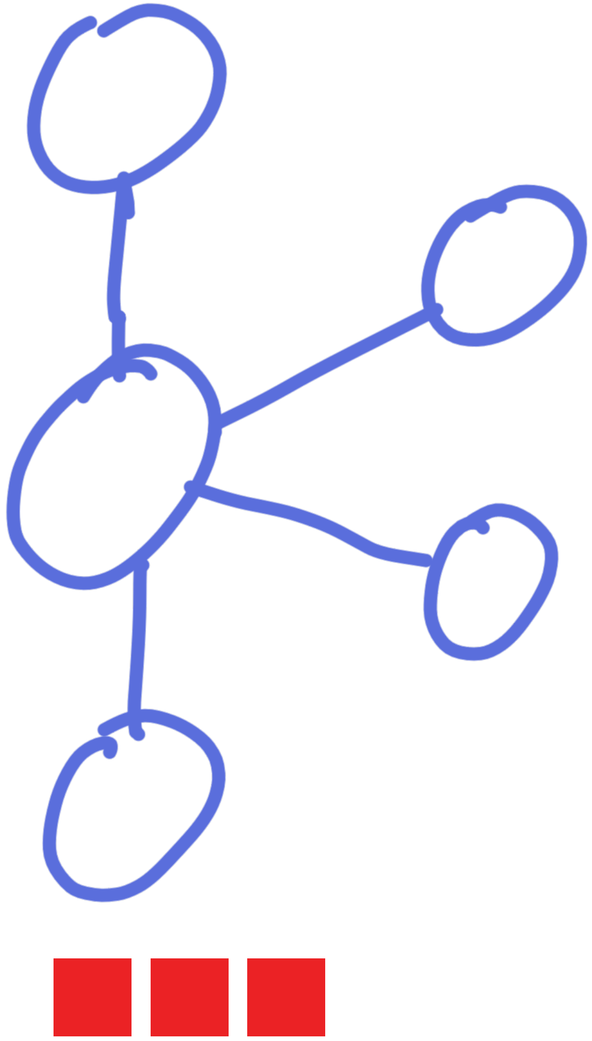



```
SELECT *
FROM   my_table
WHERE  ts_col >
          previous ts
```

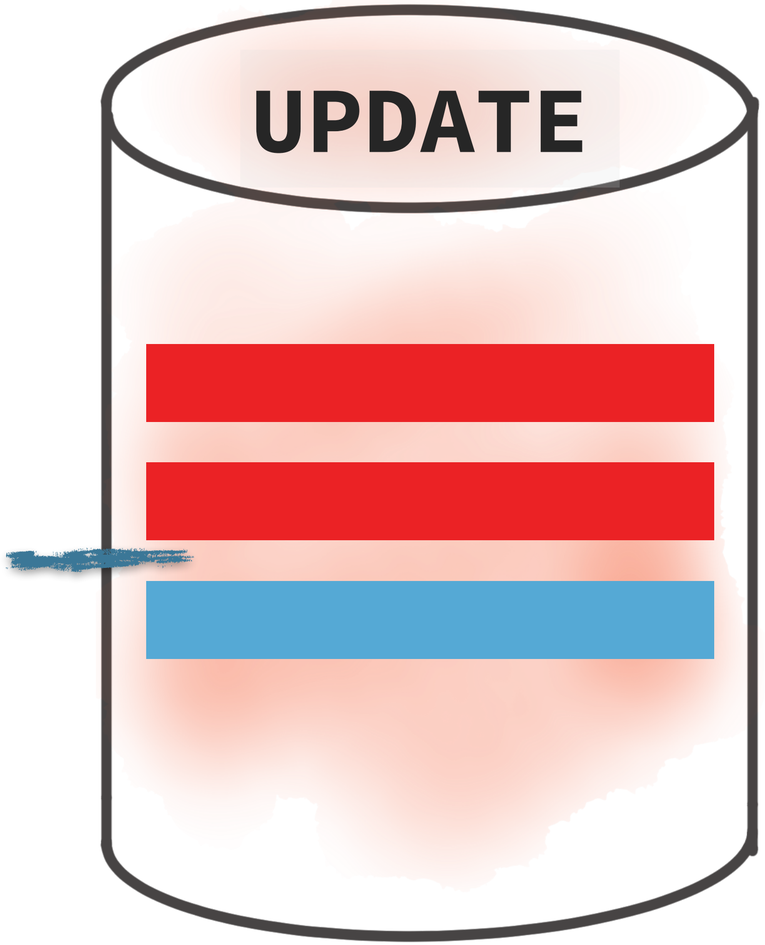


Query-based CDC

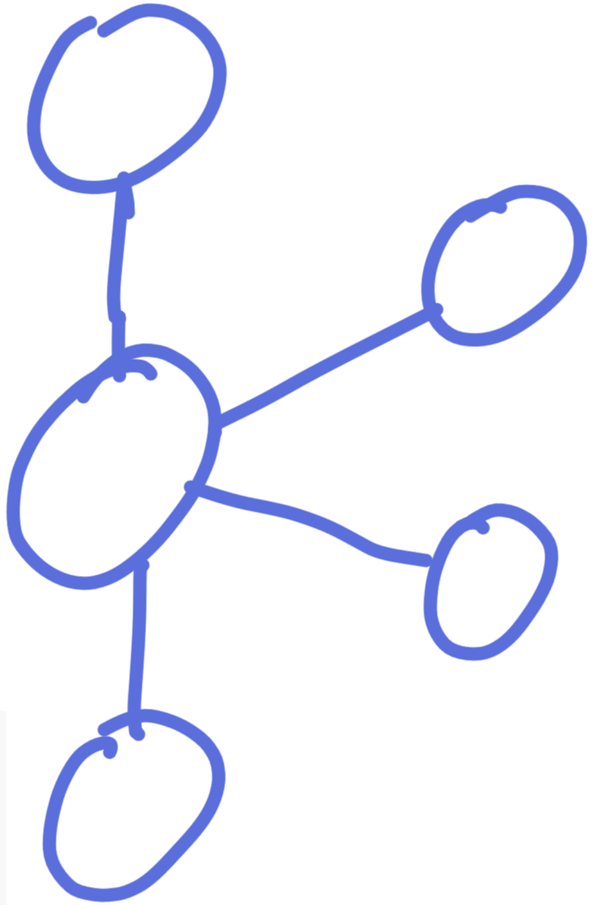




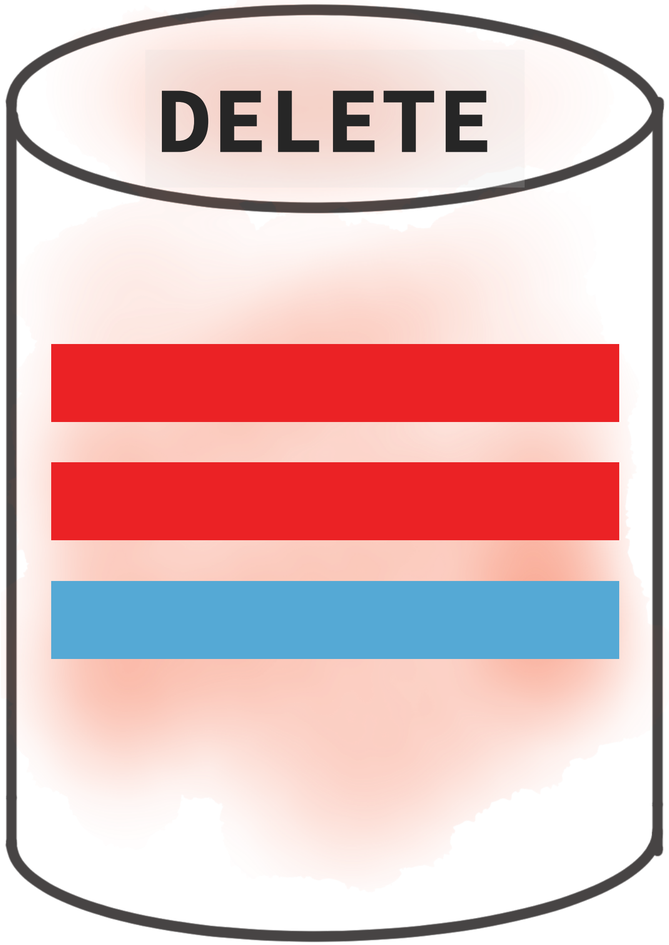
Query-based CDC

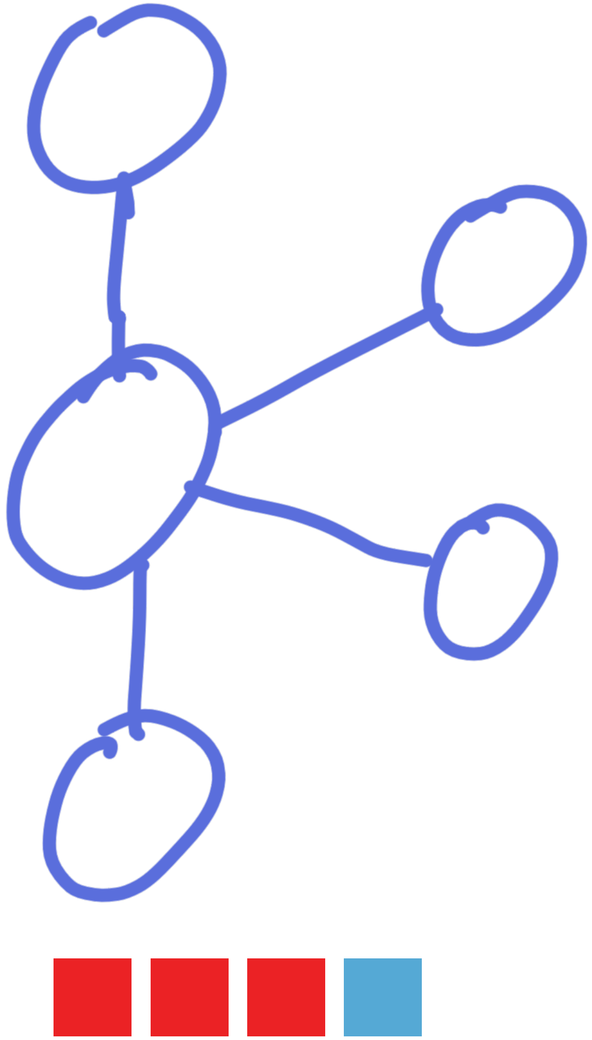



```
SELECT *
FROM   my_table
WHERE  ts_col >
         previous ts
```

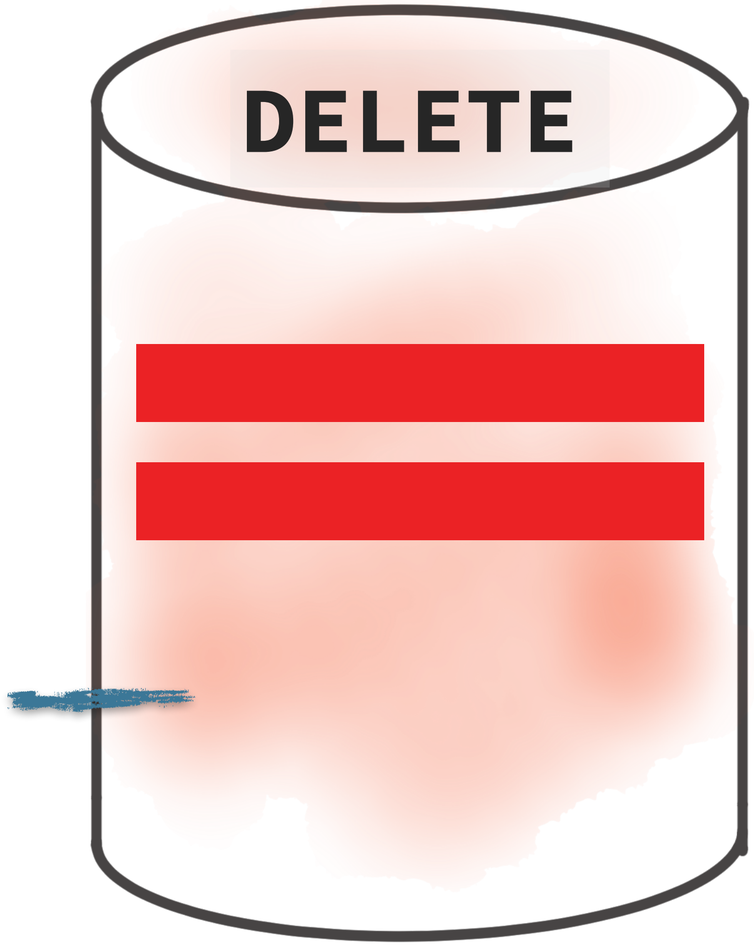


Query-based CDC

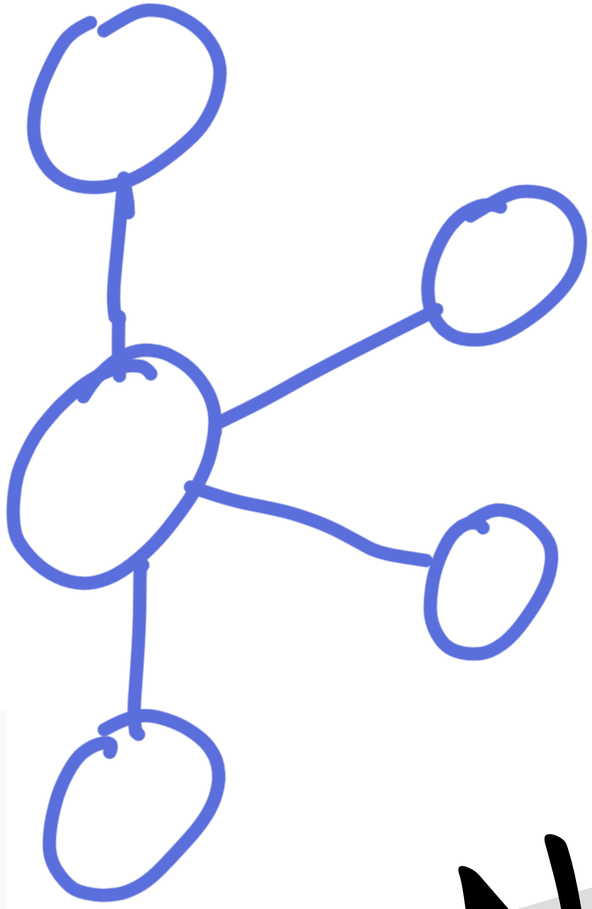




Query-based CDC




```
SELECT *
FROM   my_table
WHERE  ts_col >
         previous ts
```



Nope

Query-based CDC

orderID	status	address	updateTS

```
SELECT *
FROM   orders
WHERE  updateTS >
         previous ts
```

Query-based CDC

orderID	status	address	updateTS
42	SHIPPED	29 Acacia Road	10:54:29

```
SELECT *
FROM   orders
WHERE  updateTS >
         previous ts
```

```
{
  "orderID": 42,
  "status": "SHIPPED",
  "address": "29 Acacia Road",
  "updateTS": "10:54:29"
}
```

Query-based CDC

orderID	status	address	updateTS
42	PENDING	--	10:54:00

```
INSERT INTO orders (orderID, status)
VALUES (42, 'PENDING');
```

Query-based CDC

orderID	status	address	updateTS
42	PENDING	1640 Riverside Drive	10:54:20

```
UPDATE orders
SET address = '1640 Riverside Drive'
WHERE orderID = 42;
```

Query-based CDC

orderID	status	address	updateTS
42	PENDING	29 Acacia Road	10:54:25

```
UPDATE orders
SET address = '29 Acacia Road'
WHERE orderID = 42;
```

Query-based CDC

orderID	status	address	updateTS
42	SHIPPED	29 Acacia Road	10:54:29

```
UPDATE orders
SET status = 'SHIPPED'
WHERE orderID = 42;
```

Query-based CDC

orderID	status	address	updateTS
42	PENDING	--	10:54:00
42	PENDING	1640 Riverside Drive	10:54:20
42	PENDING	29 Acacia Road	10:54:25
42	SHIPPED	29 Acacia Road	10:54:29

Query-based CDC

orderID	status	address	updateTS
42	SHIPPED	29 Acacia Road	10:54:29

```
SELECT *
FROM   orders
WHERE  updateTS >
         previous ts
```

```
{
  "orderID": 42,
  "status": "SHIPPED",
  "address": "29 Acacia Road",
  "updateTS": "10:54:29"
}
```

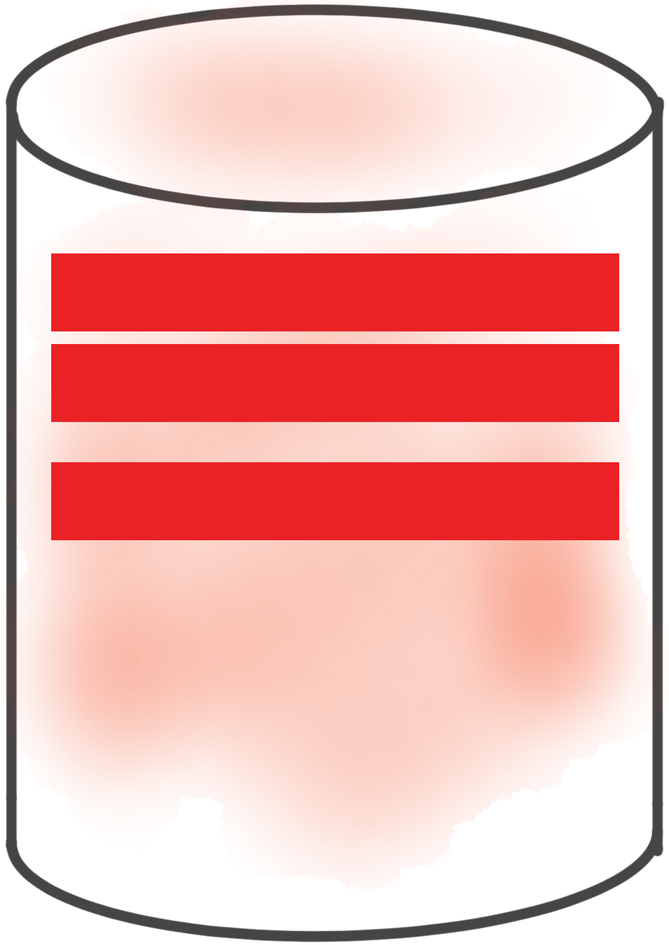
@rmoff
Event-driven
Log-based
Query-based

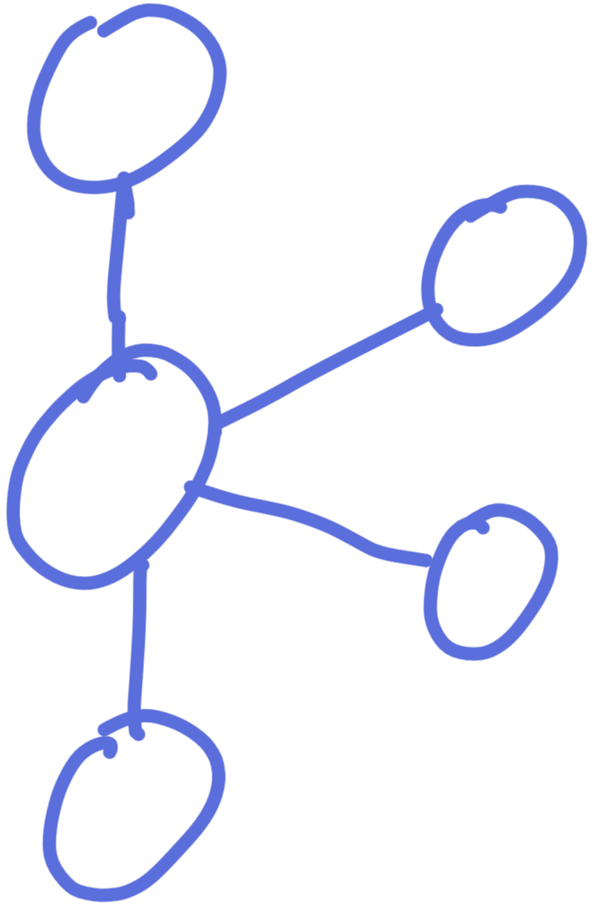
Log-based CDC

- ✅ Greater data fidelity
- ✅ Lower latency
- ✅ Lower impact on source
- 🛑 More setup steps
- 🛑 Higher system privileges required
- 🛑 For propriatory databases, usually $$$

Read more: http://cnfl.io/kafka-cdc

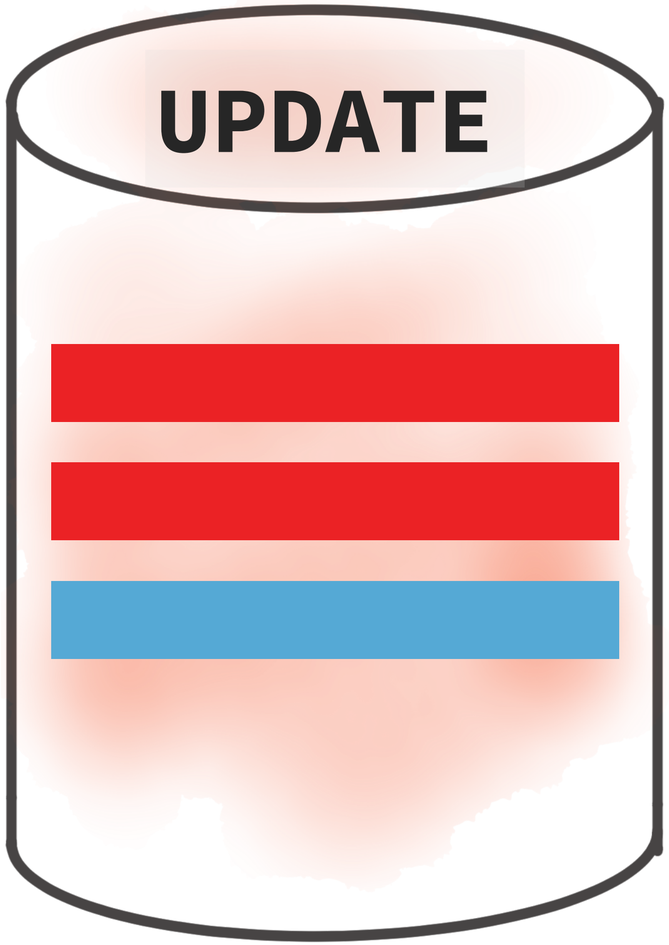
Log-based CDC



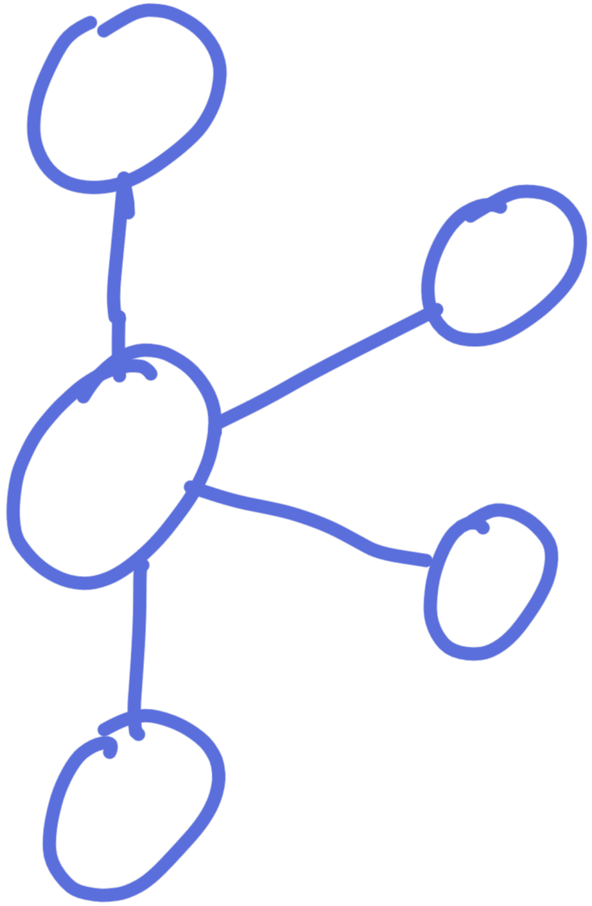


Transaction log

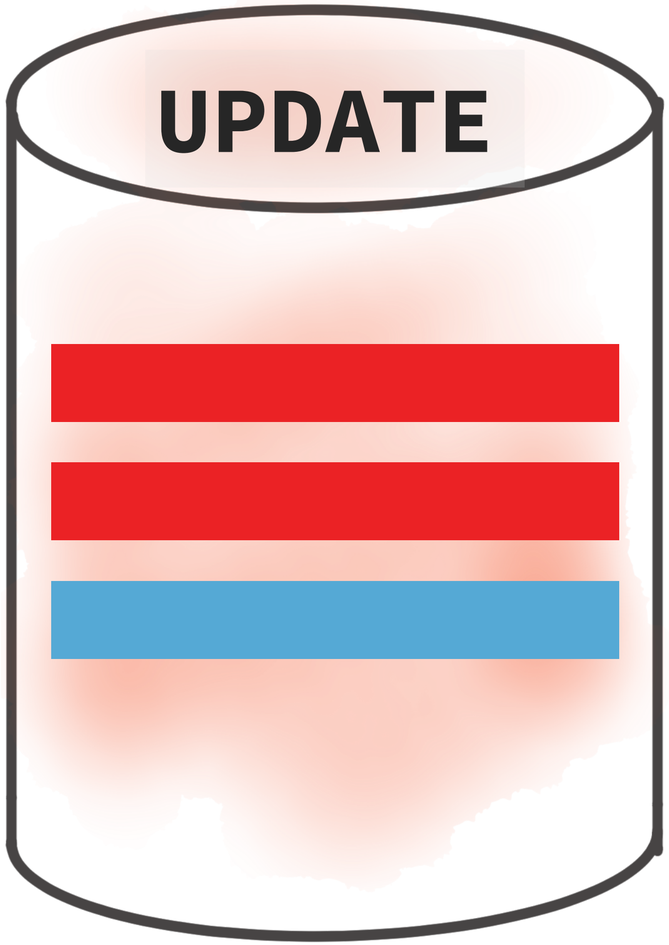
Log-based CDC



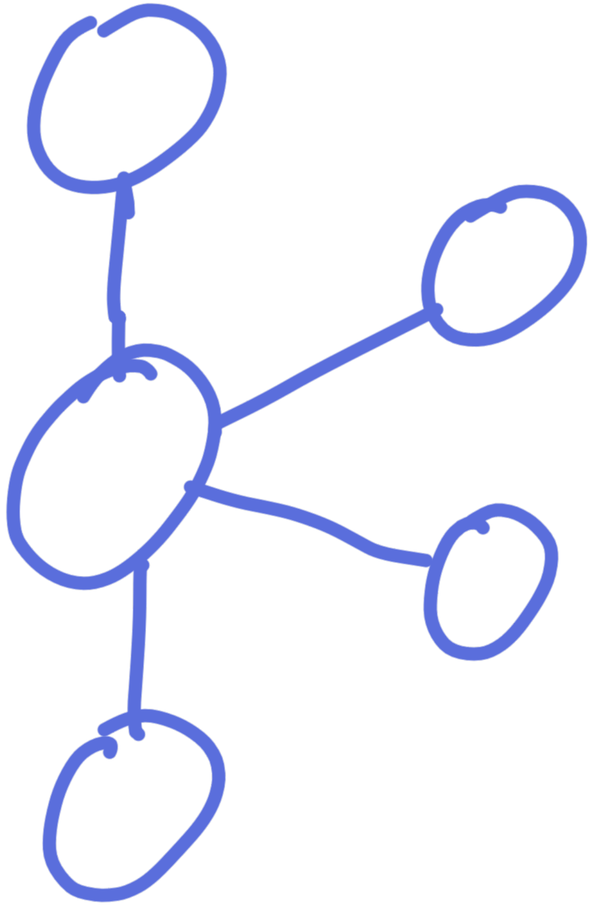
Transaction log



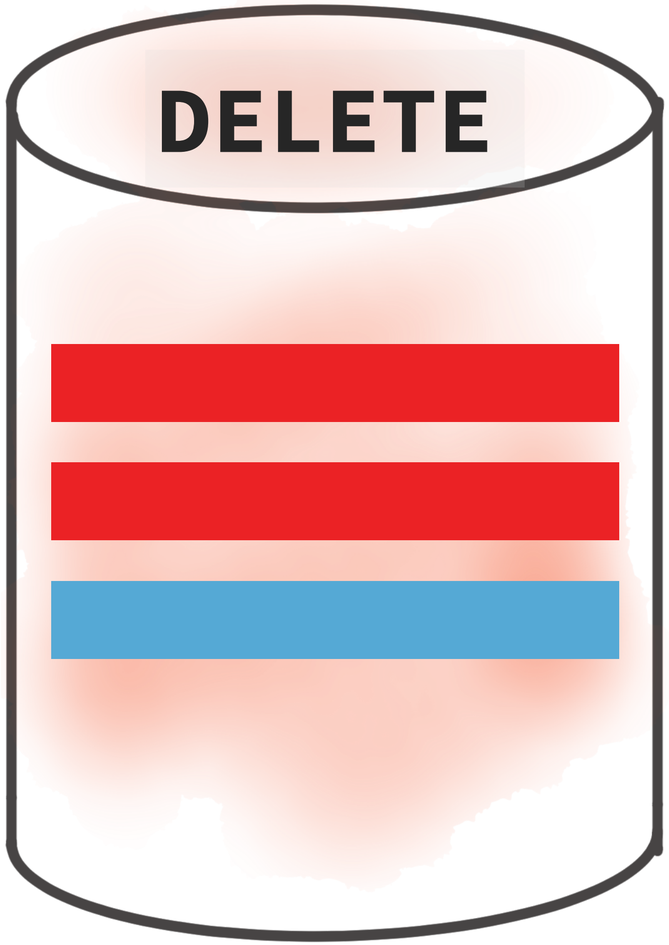
Log-based CDC



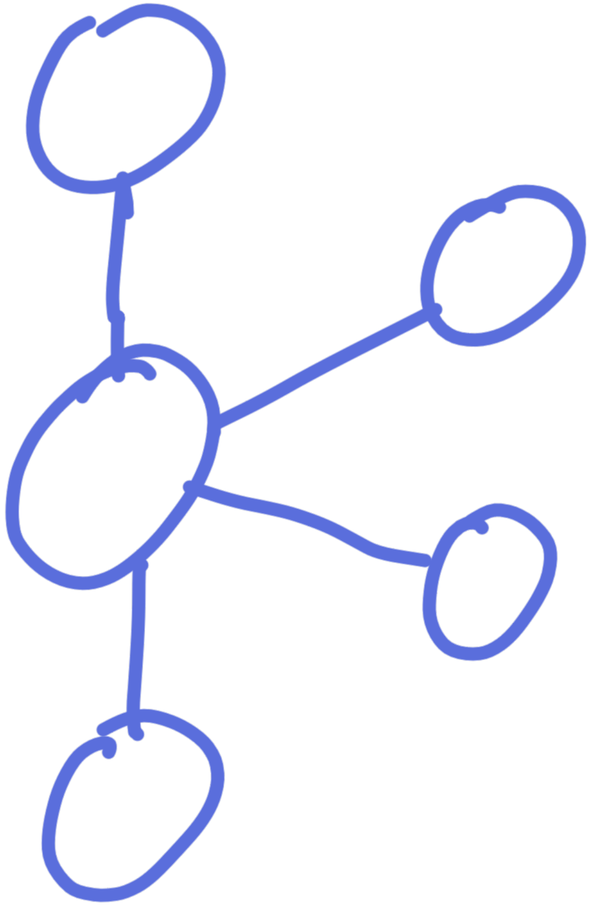
Transaction log



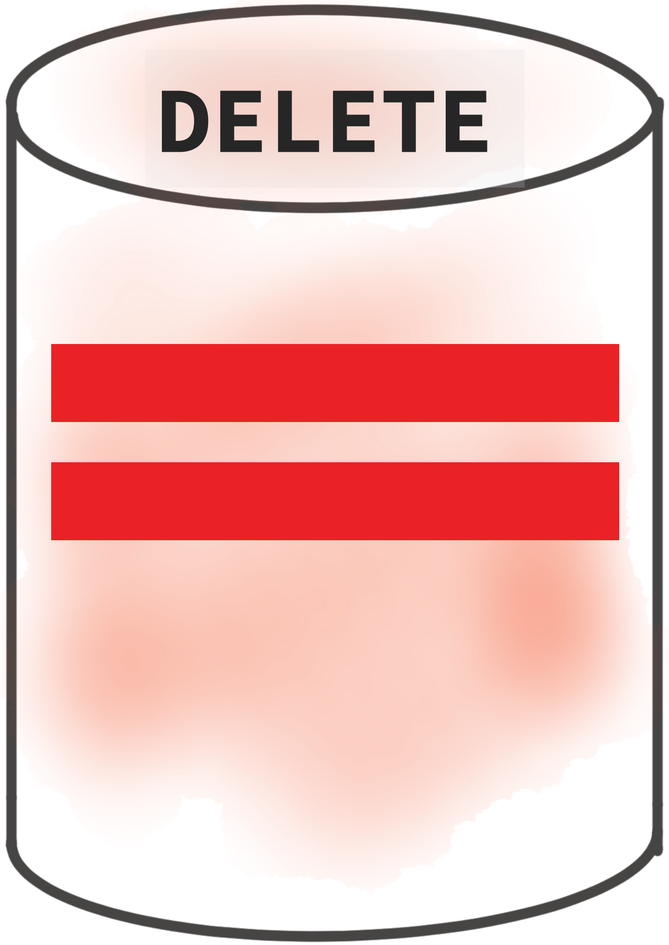
Log-based CDC



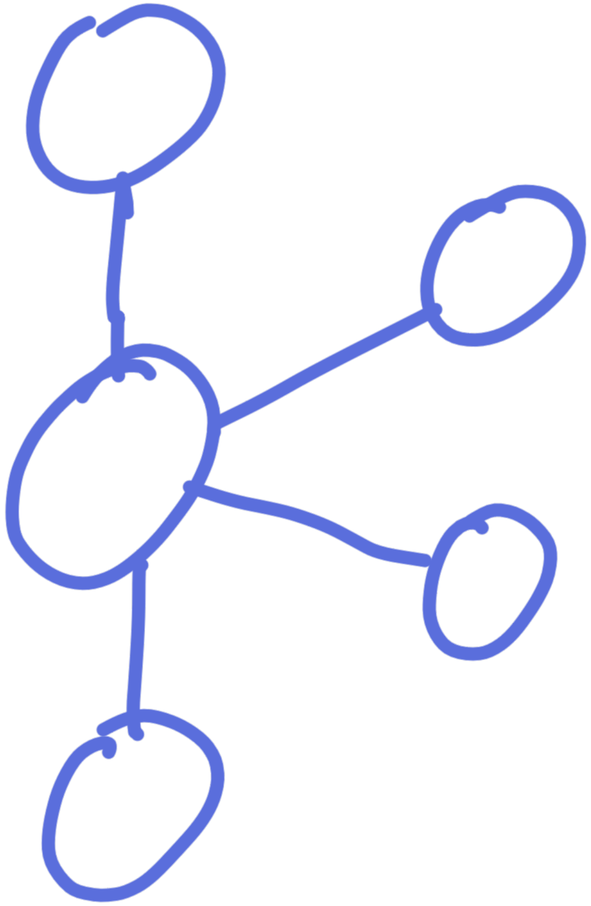
Transaction log



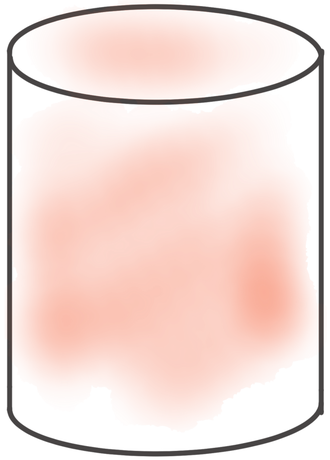
Log-based CDC

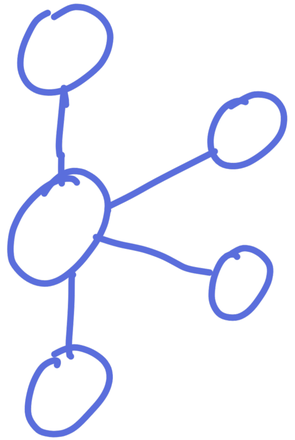


Transaction log



Log-based CDC

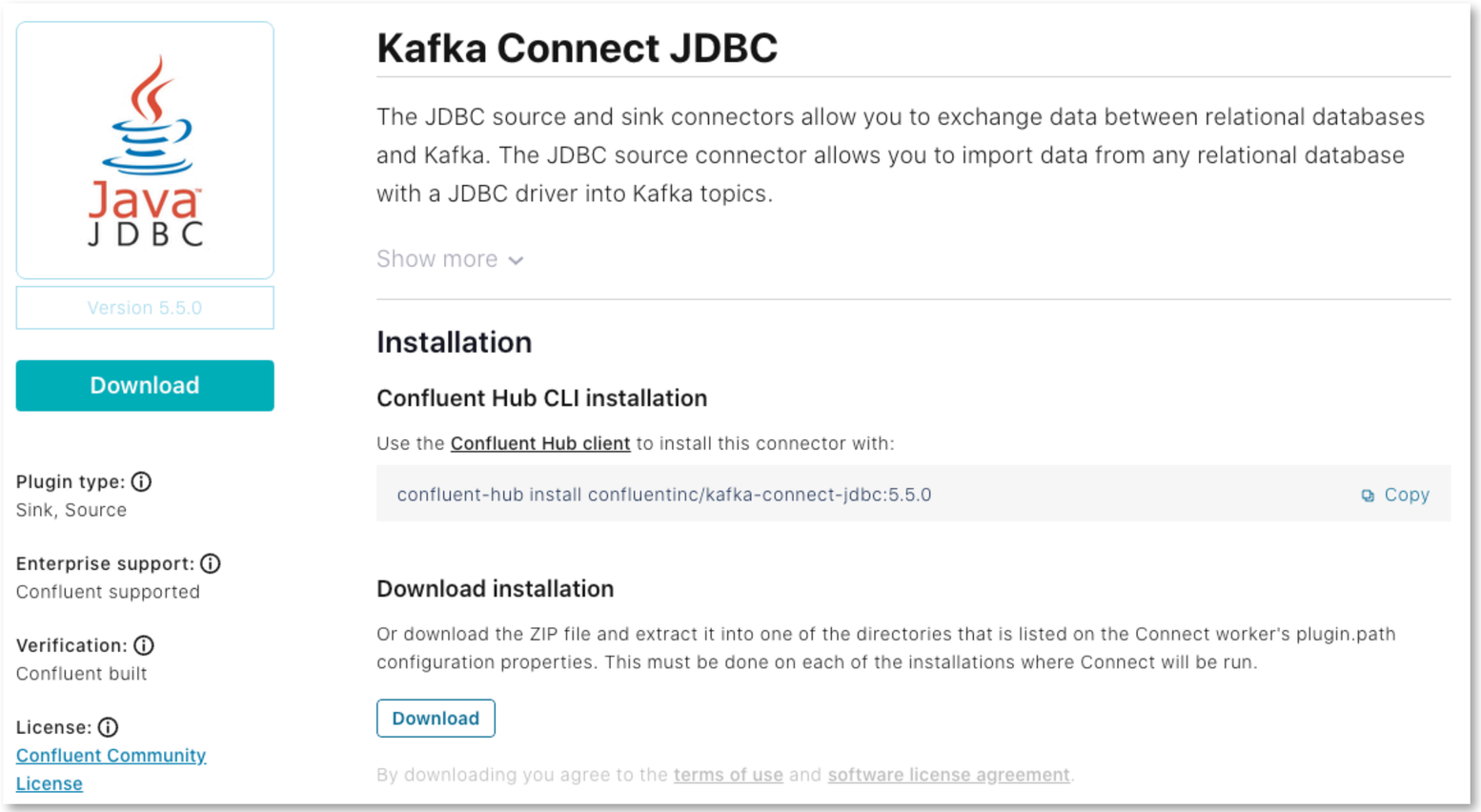




Immutable event log

tl;dr : which tool do I use?

- If you're wanting to do query-based CDC, the choice is simple:



confluent.io/hub

Which Log-Based CDC Tool?

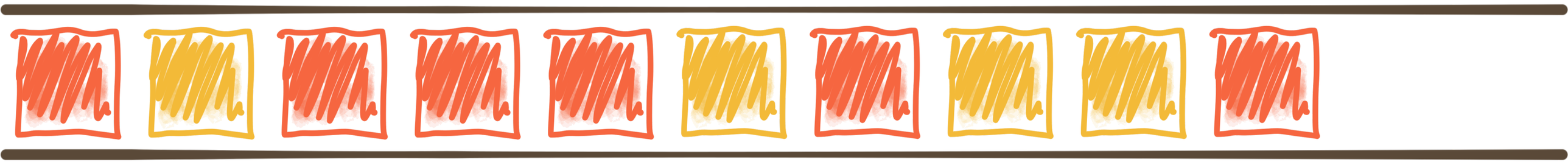
- Open Source RDBMS,
  e.g. **MySQL**, **PostgreSQL**
  - Debezium
  - (+ paid options)

- Mainframe
  e.g. **VSAM**, **IMS**
  - Attunity
  - SQData
  - tcVISION

- Proprietory RDBMS,
  e.g. **Oracle**, **MS SQL, DB2**
  - Oracle GoldenGate
    - Debezium + XStream
  - Attunity
  - IBM InfoSphere Data Replication
  - SQData
  - HVR
  - tcVISION

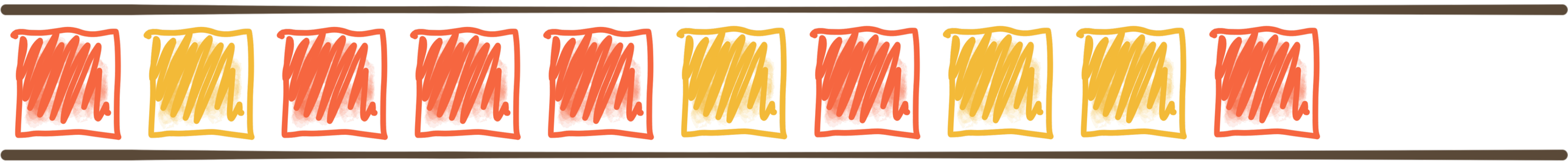
Photo by Franck V. on Unsplash

Streams of events



Time

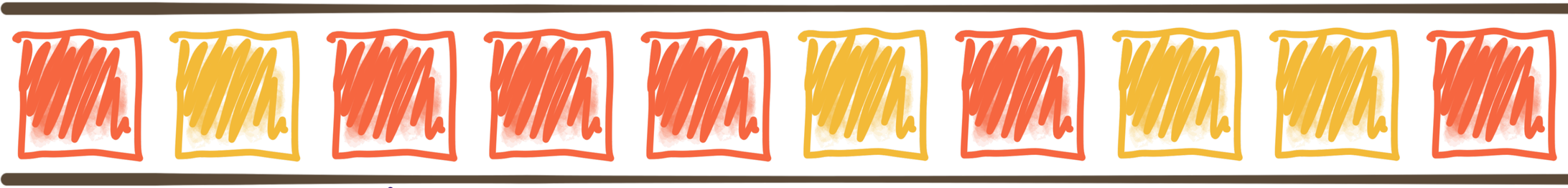
Stream Processing



Stream: widgets

Stream: widgets_red

Stream Processing with ksqlDB



Stream:
widgets

ksqlDB

```
CREATE STREAM widgets_red AS
SELECT * FROM widgets
WHERE colour='RED';
```

Stream:
widgets_red

Photo by Franck V. on Unsplash

```
SELECT *
FROM WIDGETS
WHERE WEIGHT_G > 120
```

```
SELECT COUNT(*)
FROM WIDGETS
GROUP BY PRODUCTION_LINE
```

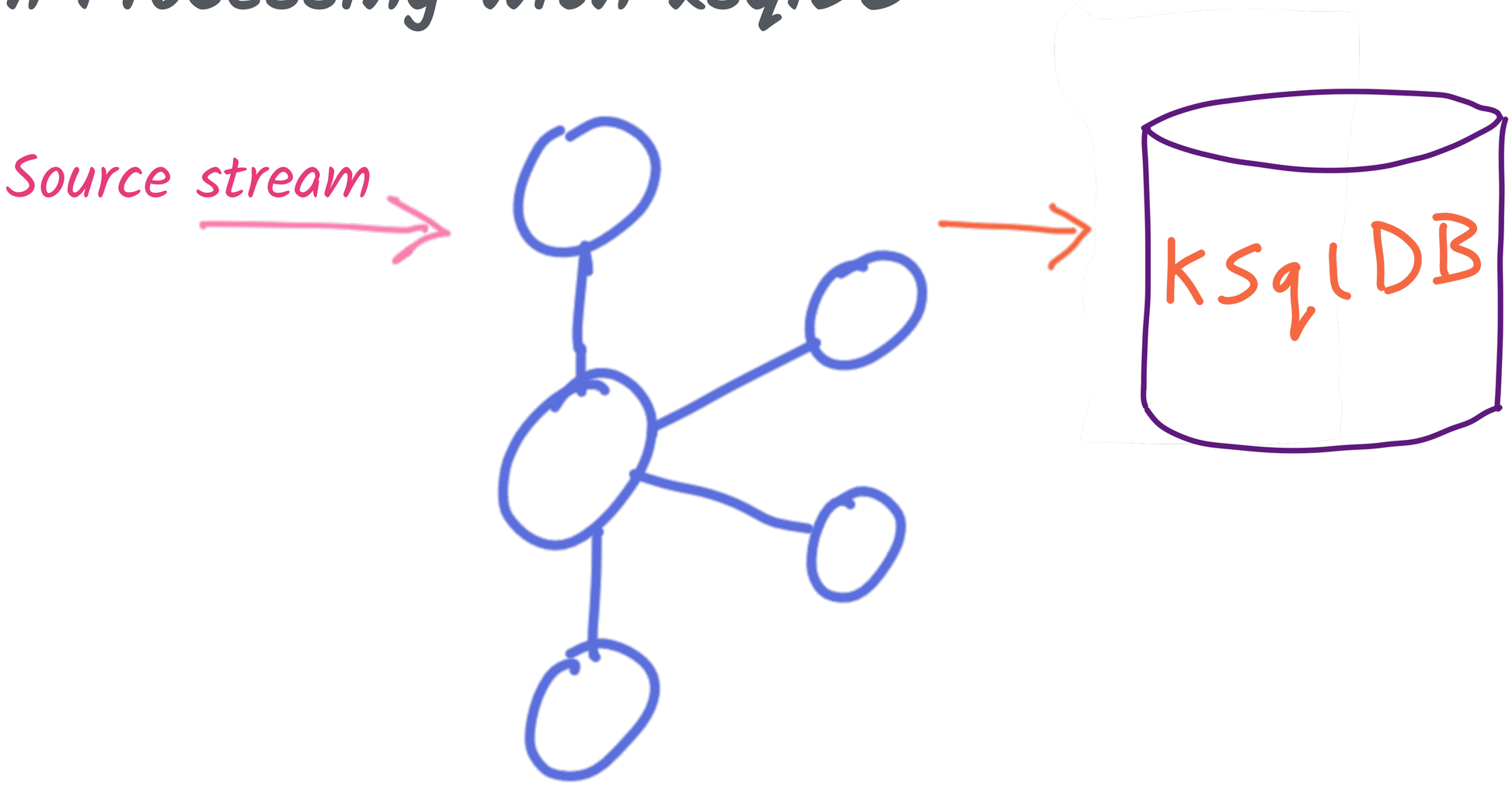
```
SELECT AVG(TEMP_CELCIUS) AS TEMP
FROM WIDGETS
GROUP BY SENSOR_ID
HAVING TEMP>20
```

Object store, data warehouse, RDBMS

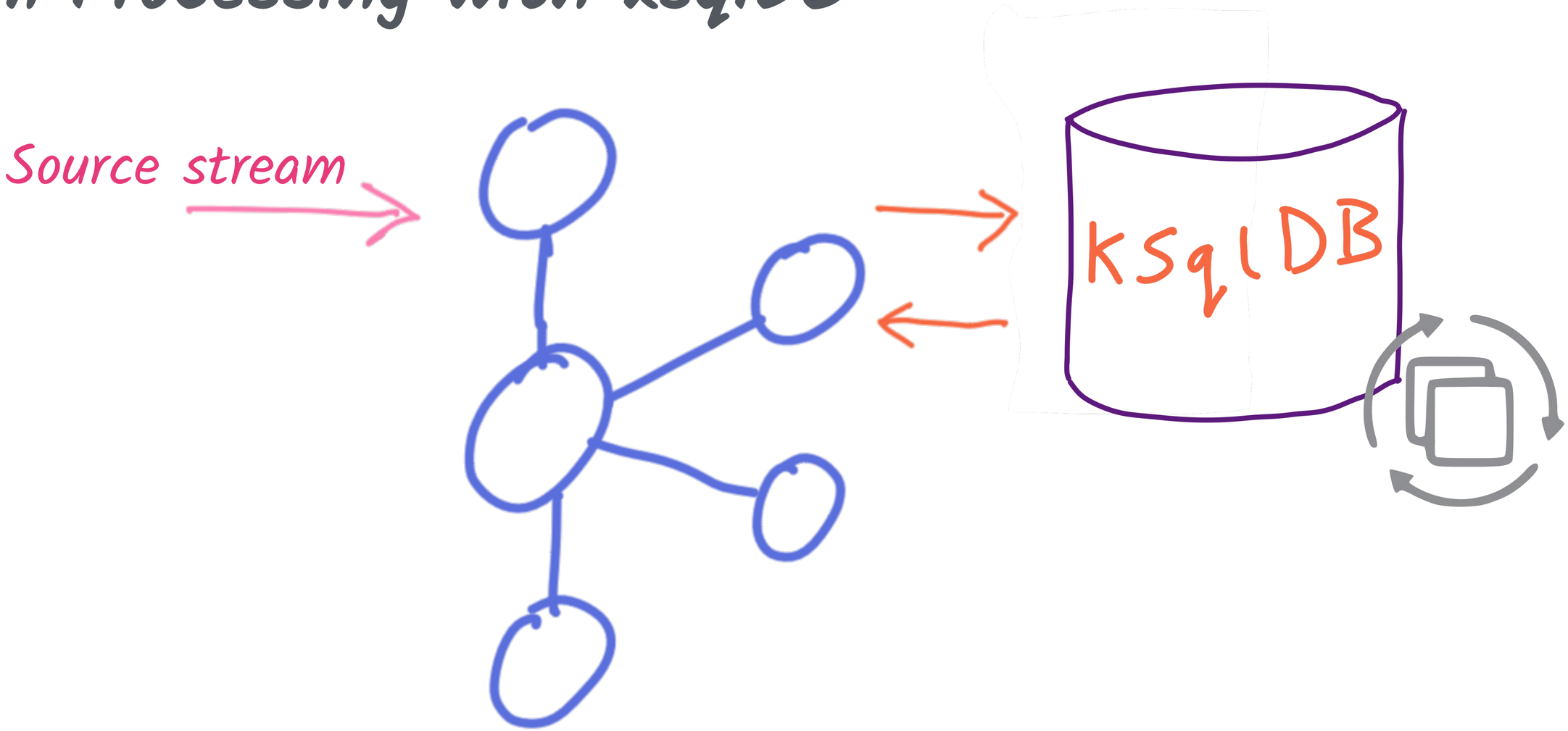
```
CREATE SINK CONNECTOR dw WITH (
'connector.class' = 'S3Connector',
'topics' = 'widgets'
…);
```


Photo by Franck V. on Unsplash

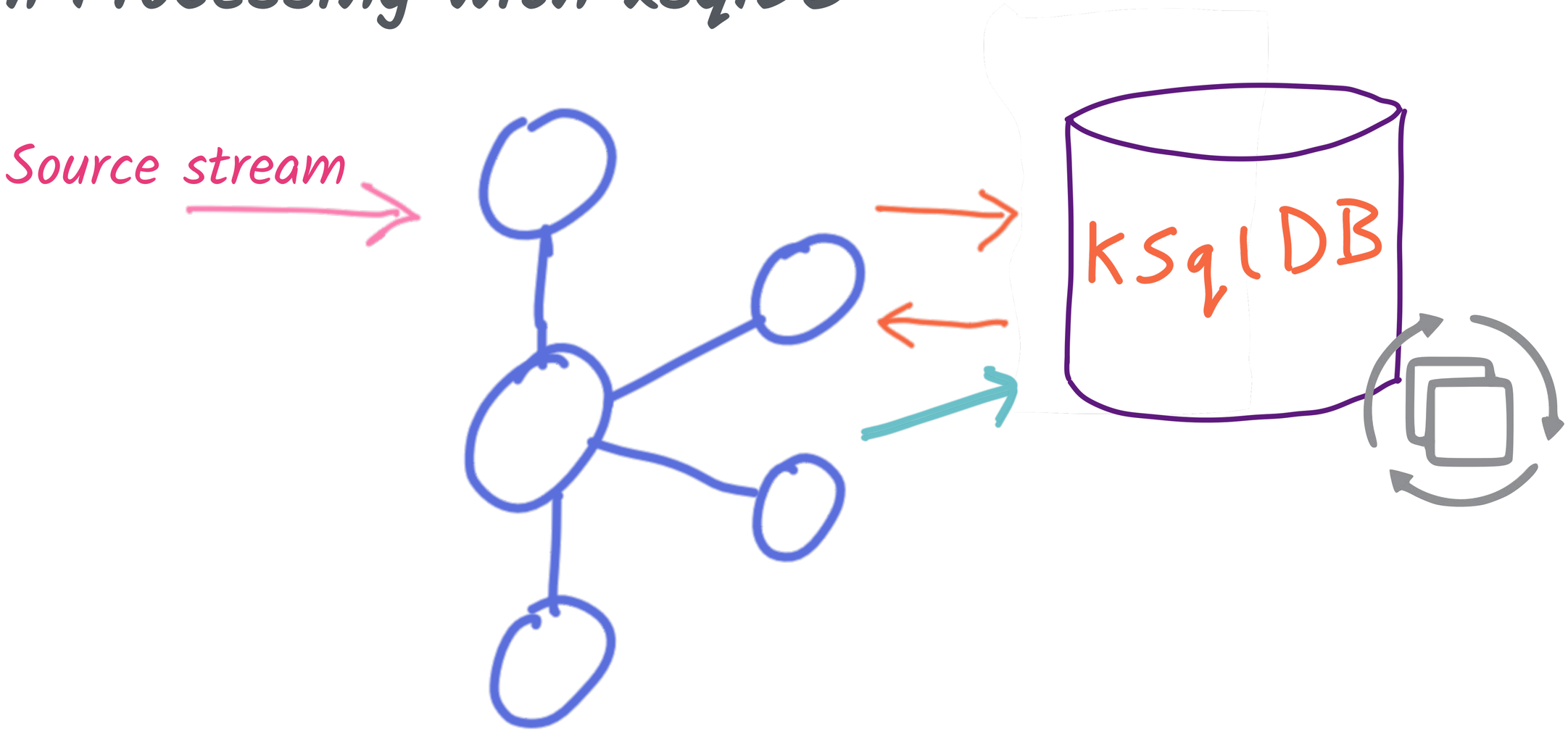
Stream Processing with ksqlDB



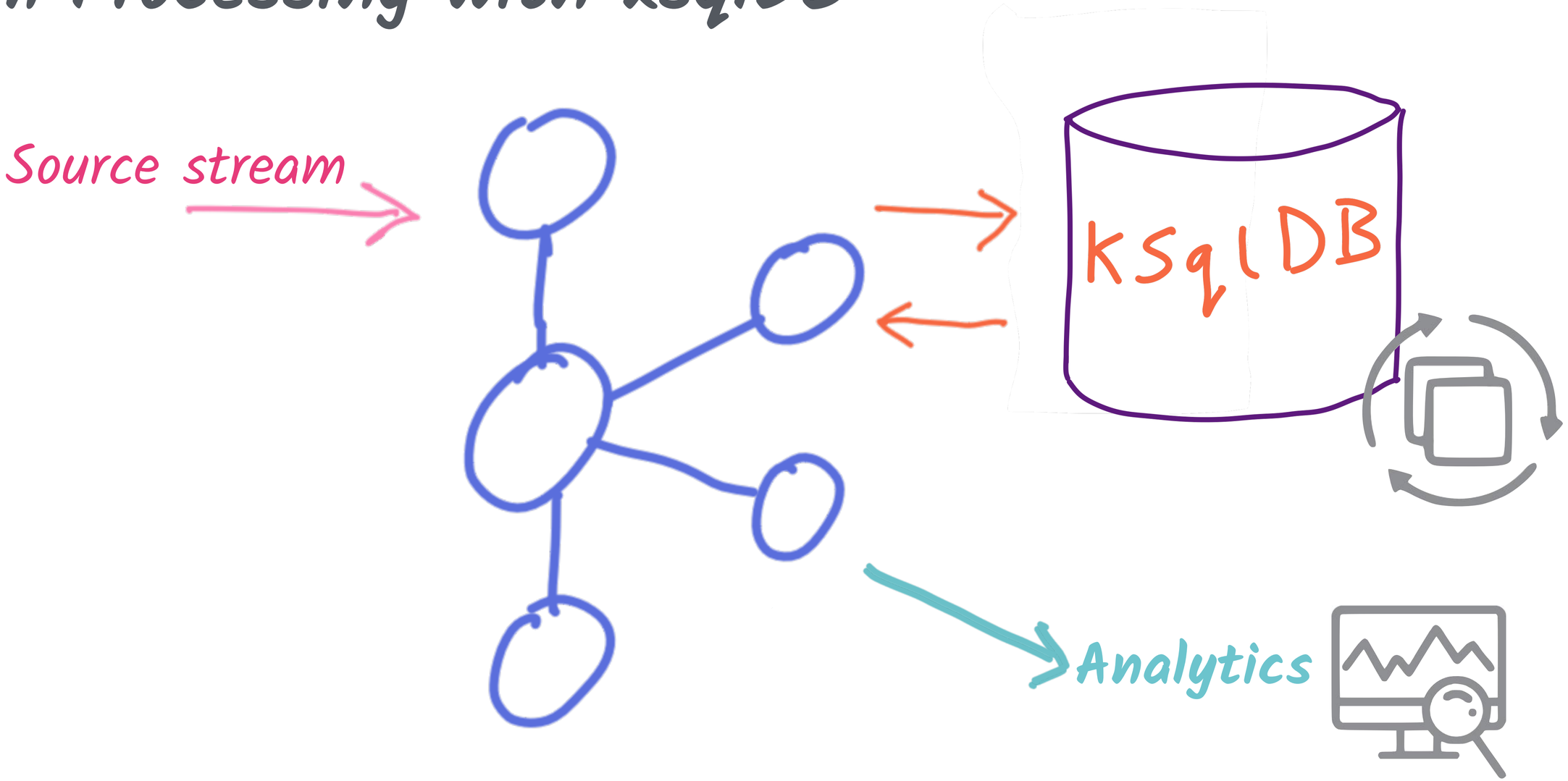
Stream Processing with ksqlDB



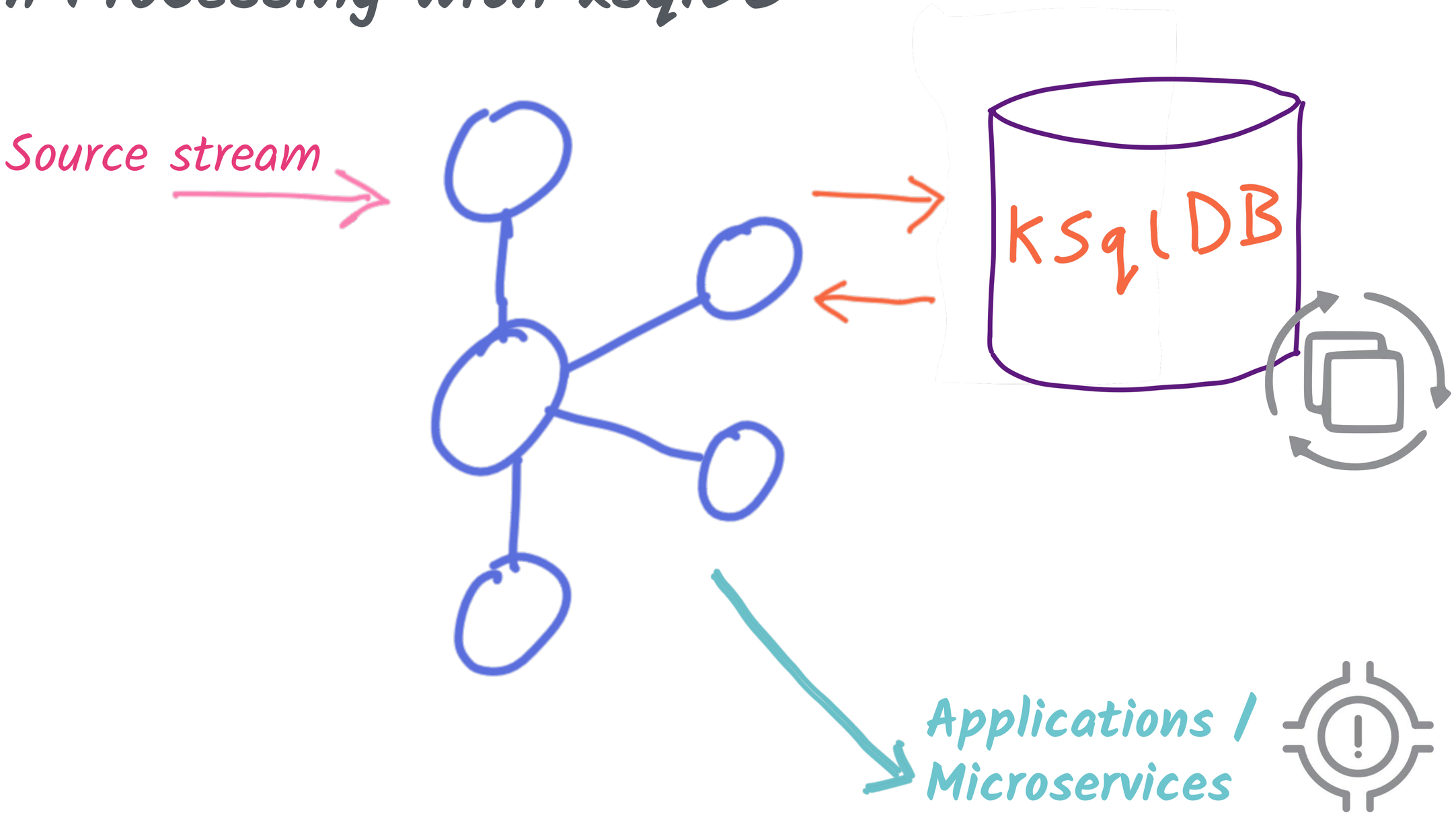
Stream Processing with ksqlDB



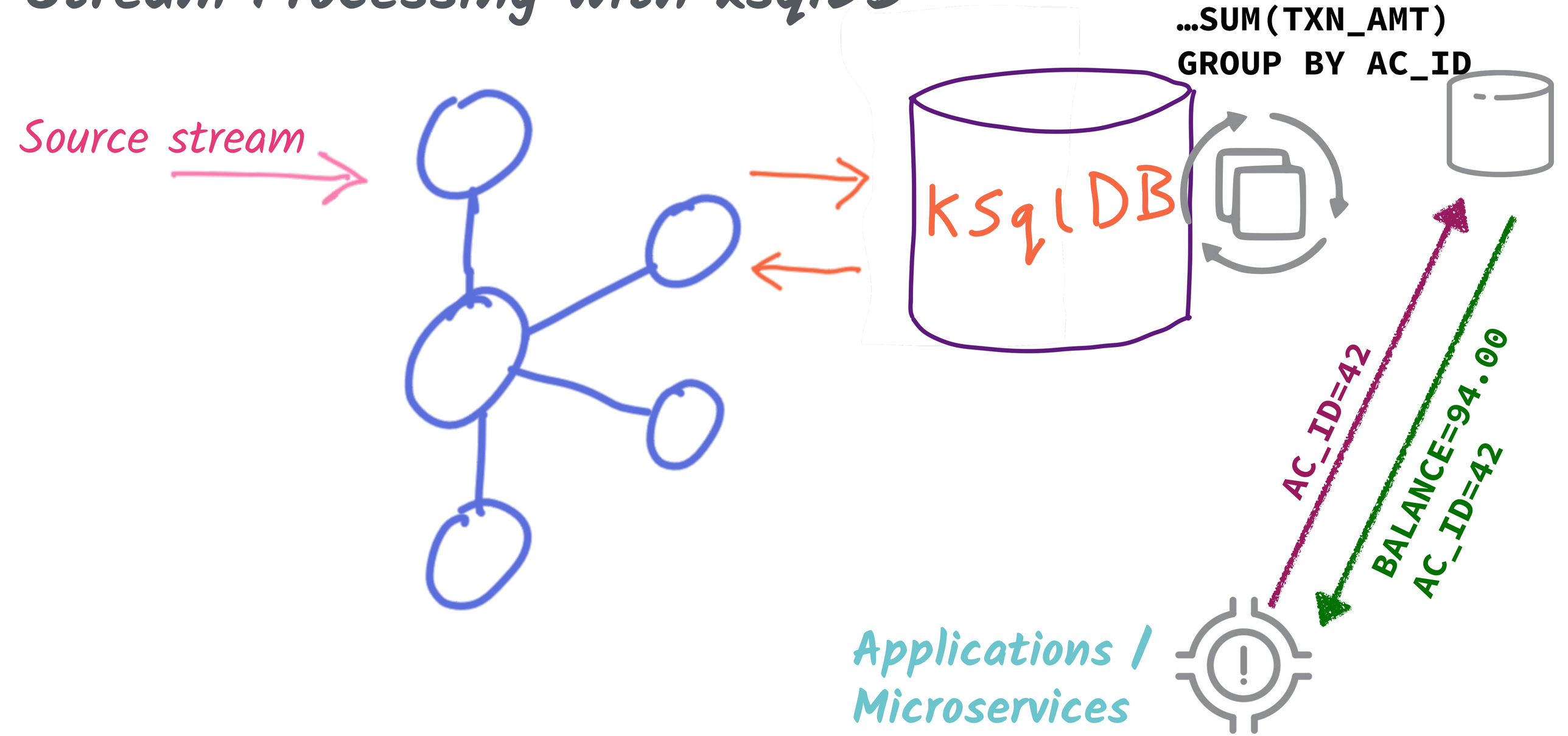
Stream Processing with ksqlDB



Stream Processing with ksqlDB



Stream Processing with ksqlDB



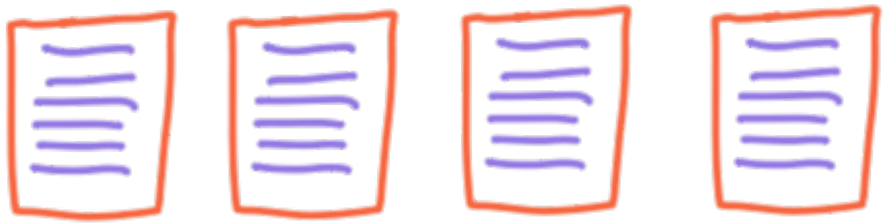
Lookups and Joins with ksqlDB

ORDERS


```
{"ordertime": 1560070133853,
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5}
```

Lookups and Joins with ksqlDB

ITEMS



ORDERS


```
{
  "id": "Item_9",
  "make": "Boyle-McDermott",
  "model": "Apiaceae",
 "unit_cost": 19.9
}
{"ordertime": 1560070133853,
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5}
```

Lookups and Joins with ksqlDB

ITEMS

ksqlDB

ORDERS

```
CREATE STREAM ORDERS_ENRICHED AS
SELECT O.*, I.*,
  O.ORDERUNITS * I.UNIT_COST
    AS TOTAL_ORDER_VALUE,
  FROM ORDERS O
       INNER JOIN ITEMS I
       ON O.ITEMID = I.ID ;
```


```
{
  "id": "Item_9",
  "make": "Boyle-McDermott",
  "model": "Apiaceae",
 "unit_cost": 19.9
}
{"ordertime": 1560070133853,
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5}
```

Lookups and Joins with ksqlDB

ITEMS

ORDERS

```
CREATE STREAM ORDERS_ENRICHED AS
SELECT O.*, I.*,
  O.ORDERUNITS * I.UNIT_COST
    AS TOTAL_ORDER_VALUE,
  FROM ORDERS O
       INNER JOIN ITEMS I
       ON O.ITEMID = I.ID ;
```

ksqlDB

ORDERS_ENRICHED


```
{
  "id": "Item_9",
  "make": "Boyle-McDermott",
  "model": "Apiaceae",
 "unit_cost": 19.9
}
{"ordertime": 1560070133853,
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5}
```

```
{
  "ordertime": 1560070133853,
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5,
  "make": "Boyle-McDermott",
  "model": "Apiaceae",
  "unit_cost": 19.9,
  "total_order_value": 99.5
}
```

Streams & Tables

CONFLUENT

Streams and Tables

Kafka topic (k/v bytes)

```
{ "event_ts": "2020-02-17T15:22:00Z",
  "person"  : "robin",
  "location": "Leeds"
}
```

```
{ "event_ts": "2020-02-17T17:23:00Z",
  "person"  : "robin",
  "location": "London"
}
```

```
{ "event_ts": "2020-02-17T22:23:00Z",
  "person"  : "robin",
  "location": "Wakefield"
}
```

```
{ "event_ts": "2020-02-18T09:00:00Z",
  "person"  : "robin",
  "location": "Leeds"
}
```

ksqlDB Stream

```
+--------------------+-------+---------+
|EVENT_TS            |PERSON |LOCATION |
+--------------------+-------+---------+
|2020-02-17 15:22:00 |robin  |Leeds    |
|2020-02-17 17:23:00 |robin  |London   |
|2020-02-17 22:23:00 |robin  |Wakefield|
|2020-02-18 09:00:00 |robin  |Leeds    |
```

Stream: Topic + Schema

ksqlDB Table

```
+-------+---------+
|PERSON |LOCATION |
+-------+---------+
|robin  |Leeds    |
```

Table: **state** for given **key**

Topic + Schema

Stateful aggregations in ksqlDB

Kafka topic

```
{ "event_ts": "2020-02-17T15:22:00Z",
  "person"  : "robin",
  "location": "Leeds"
}
```

```
{ "event_ts": "2020-02-17T17:23:00Z",
  "person"  : "robin",
  "location": "London"
}
```

```
{ "event_ts": "2020-02-17T22:23:00Z",
  "person"  : "robin",
  "location": "Wakefield"
}
```

```
{ "event_ts": "2020-02-18T09:00:00Z",
  "person"  : "robin",
  "location": "Leeds"
}
```

```
SELECT PERSON,
       COUNT(*)
FROM   MOVEMENTS
GROUP BY PERSON;
```

```
+-------+------------------+
|PERSON | LOCATION_CHANGES |
+-------+------------------+
|robin  | 4                |
```

```
SELECT PERSON,
       COUNT_DISTINCT(LOCATION)
FROM   MOVEMENTS
GROUP BY PERSON;
```

```
+-------+------------------+
|PERSON | UNIQUE_LOCATIONS |
+-------+------------------+
|robin  | 3                |
```

Aggregations can be across the entire input, or windowed (TUMBLING, HOPPING, SESSION)

Stateful aggregations in ksqlDB

Kafka topic

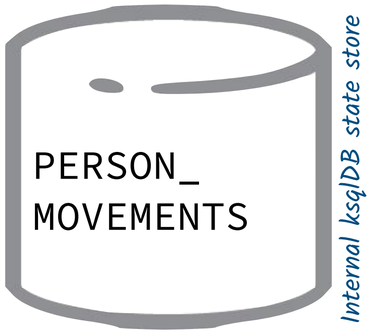
```
{ "event_ts": "2020-02-17T15:22:00Z",
  "person"  : "robin",
  "location": "Leeds"
}
```

```
{ "event_ts": "2020-02-17T17:23:00Z",
  "person"  : "robin",
  "location": "London"
}
```

```
{ "event_ts": "2020-02-17T22:23:00Z",
  "person"  : "robin",
  "location": "Wakefield"
}
```

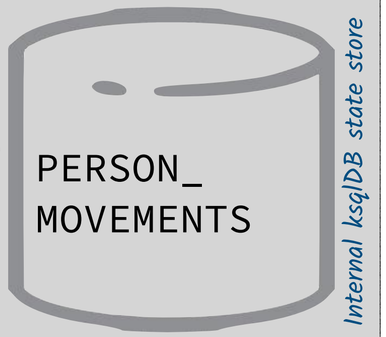
```
{ "event_ts": "2020-02-18T09:00:00Z",
  "person"  : "robin",
  "location": "Leeds"
}
```

```
CREATE TABLE PERSON_MOVEMENTS AS
 SELECT PERSON,
        COUNT_DISTINCT(LOCATION) AS
          UNIQUE_LOCATIONS,
        COUNT(*) AS LOCATION_CHANGES
   FROM MOVEMENTS
 GROUP BY PERSON;
```



Pull and Push queries in ksqlDB

```
CREATE TABLE PERSON_MOVEMENTS AS
 SELECT PERSON,
        COUNT_DISTINCT(LOCATION) AS
          UNIQUE_LOCATIONS,
        COUNT(*) AS LOCATION_CHANGES
   FROM MOVEMENTS
 GROUP BY PERSON;
```




```
ksql> SELECT LOCATION_CHANGES, UNIQUE_LOCATIONS
      FROM PERSON_MOVEMENTS
      WHERE ROWKEY='robin';
+------------------+-----------------+
|LOCATION_CHANGES  |UNIQUE_LOCATIONS |
+------------------+-----------------+
|3                 |3                |
Query terminated
ksql>
```

Pull query

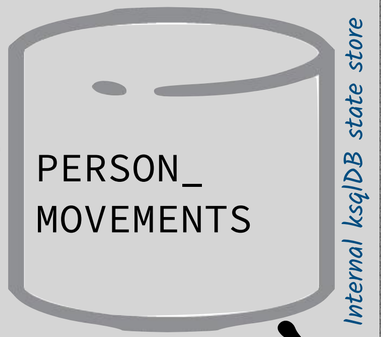
```
ksql> SELECT LOCATION_CHANGES, UNIQUE_LOCATIONS
      FROM PERSON_MOVEMENTS
      WHERE ROWKEY='robin';
      EMIT CHANGES;
+------------------+-----------------+
|LOCATION_CHANGES  |UNIQUE_LOCATIONS |
+------------------+-----------------+
|1                 |1                |
|2                 |2                |
|3                 |3                |
|4                 |3                |

Press CTRL-C to interrupt
```

Push query

ksqlDB REST API

```
CREATE TABLE PERSON_MOVEMENTS AS
 SELECT PERSON,
        COUNT_DISTINCT(LOCATION) AS
          UNIQUE_LOCATIONS,
        COUNT(*) AS LOCATION_CHANGES
   FROM MOVEMENTS
 GROUP BY PERSON;
```



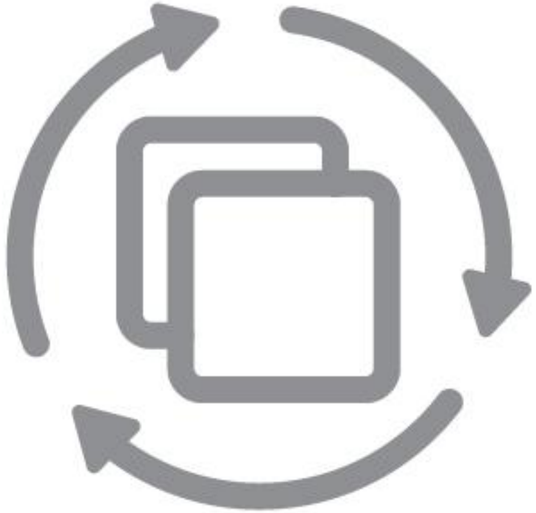

```
curl -s -X "POST" "http://localhost:8088/query" \
-H "Content-Type: application/vnd.ksql.v1+json; charset=utf-8" \
-d '{"ksql":"SELECT UNIQUE_LOCATIONS FROM PERSON_MOVEMENTS WHERE ROWKEY='\''robin'\'';"}'
```

```
{"value":"3"}
```

Pull and Push queries in ksqlDB

Pull query

Push query



Tells you: Point in time value

All value changes

Exits: Immediately

Never

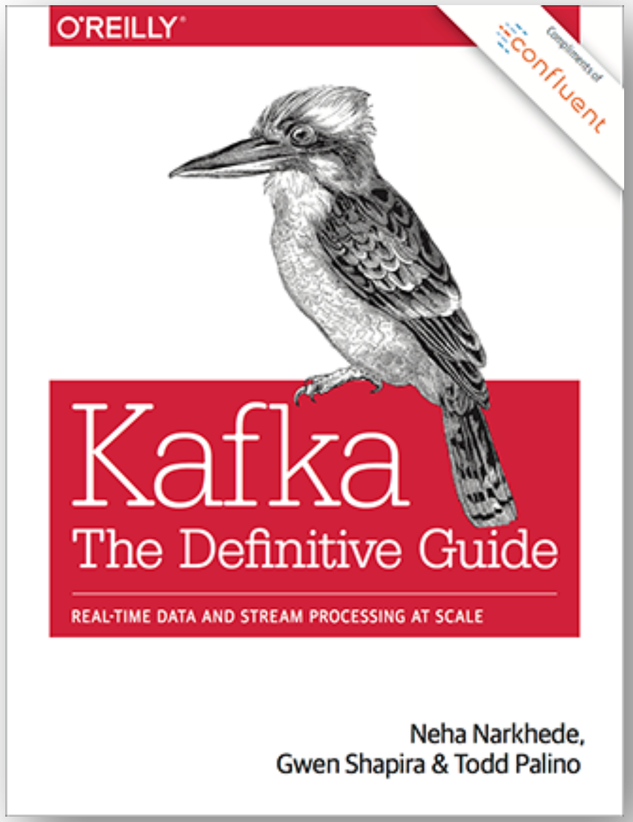
Want to learn more?

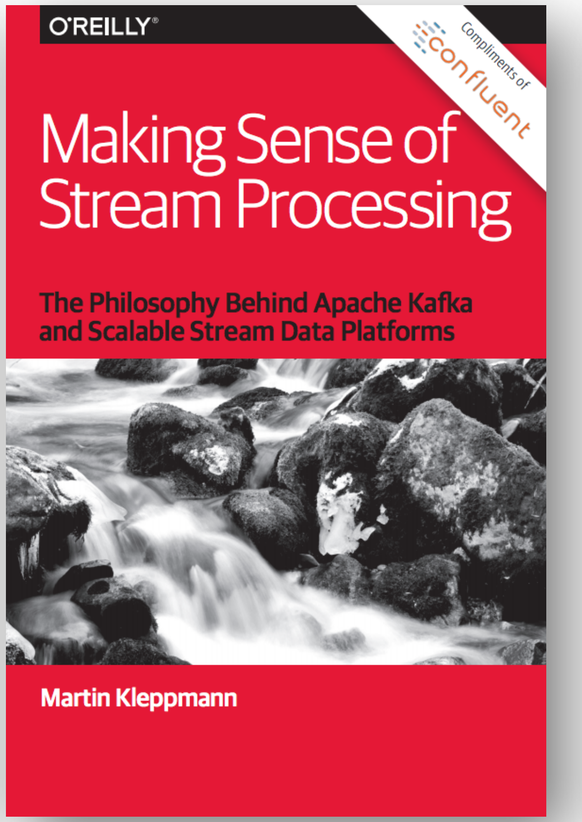
CONFLUENT

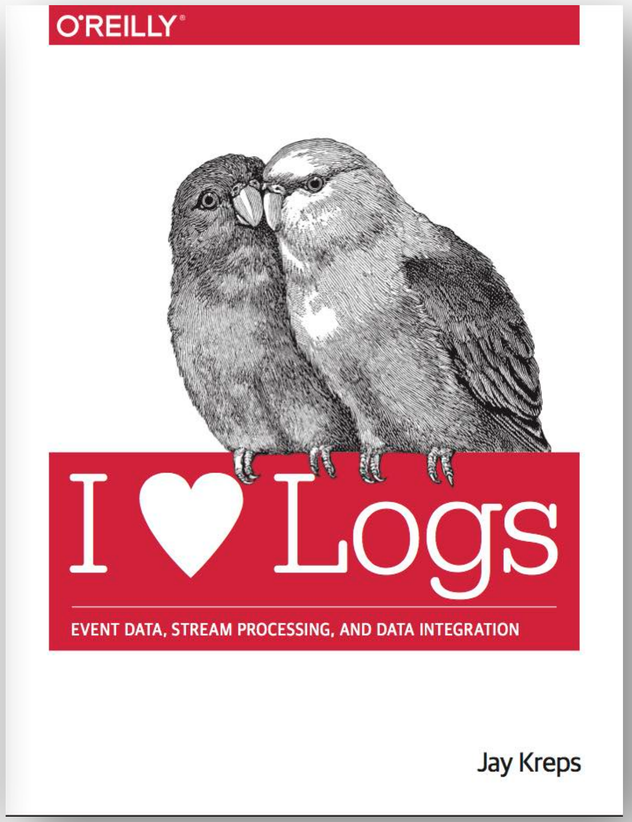
CTAs, not CATs
(sorry, not sorry)

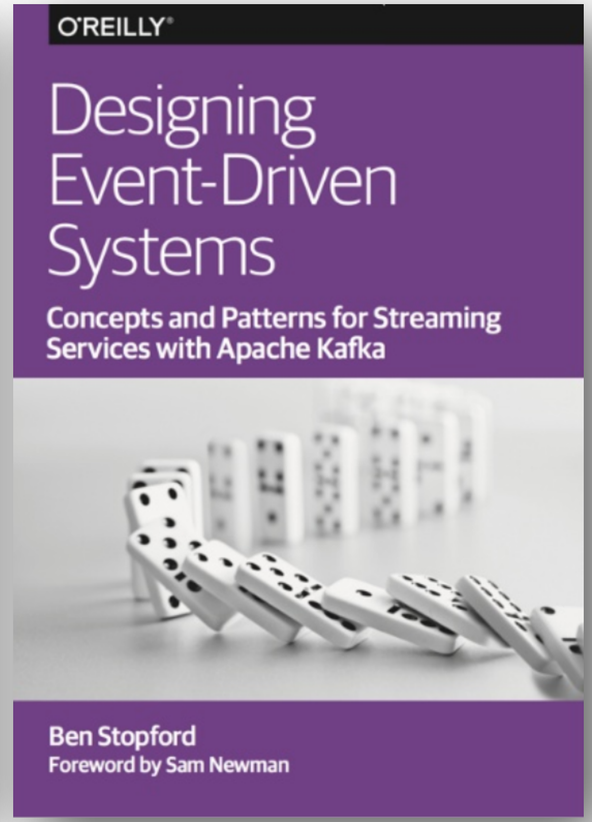
Free Books!

https://rmoff.dev/aces









60DEVADV

Free money!
(additional $60 towards your bill 😄)

$50 USD off your bill each calendar month for the first three months when you sign up

https://rmoff.dev/ccloud

confluent cloud

Fully Managed Kafka as a Service

* Limited availability. Activate by 11th September 2020. Expires after 90 days of activation. Any unused promo value on the expiration date will be forfeited.

Learn Kafka.
Start building with Apache Kafka at Confluent Developer.
developer.confluent.io

Confluent Community Slack group

cnfl.io/slack

Further reading / watching

- **Kafka as a Platform: the Ecosystem from the Ground Up**
  - https://rmoff.dev/kafka101
- **Apache Kafka and ksqlDB in Action: Let's Build a Streaming Data Pipeline!**
  - https://rmoff.dev/ljc-kafka-01
- **From Zero to Hero with Kafka Connect**
  - https://rmoff.dev/ljc-kafka-02
- **Introduction to ksqlDB**
  - https://rmoff.dev/ljc-kafka-03
- **The Changing Face of ETL: Event-Driven Architectures for Data Engineers**
  - https://rmoff.dev/oredev19-changing-face-of-etl
- **🚂On Track with Apache Kafka: Building a Streaming Platform solution with Rail Data**
  - https://rmoff.dev/oredev19-on-track-with-kafka

@rmoff #ACEsAtHome

#EOF

rmoff.dev/talks